ENERGY-EFFICIENT CIRCUITS AND SYSTEM ARCHITECTURES TO ENABLE INTELLIGENCE AT THE EDGE OF THE CLOUD

A Dissertation
Presented to
The Academic Faculty

By

Anvesha Amaravati

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Georgia Institute of Technology

Georgia Institute of Technology

December 2018

ENERGY-EFFICIENT CIRCUITS AND SYSTEM ARCHITECTURES TO ENABLE INTELLIGENCE AT THE EDGE OF THE CLOUD

Approved by:

Dr. Arijit Raychowdhury
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Justin Romberg
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Sung Kyu Lim
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Hyesoon Kim
College of Computing
Georgia Institute of Technology

Dr. Keith Bowman
Manager
Qualcomm Inc

Date Approved: December , 2018

ACKNOWLEDGEMENTS

I want to thank the support I received during my Ph.D. from all faculty members and collegues. It wouldn't have been possible to write the thesis without the constant help of my adviser Prof. Arijit Raychowdhury. I want to thank my parents and my brother for their continuous support. I got an opportunity to collaborate with Prof. Justin Romberg and Kyle Xu for compressive sensing gesture recognition project. I improved my knowledge of algorithms from Kyle. I want to thank Prof. Sung-Kyu Lim, Prof. Hyesoon Kim, Prof. Justin Romberg and Dr. Keith Bowman for agreeing to be committee members for my defense and their valuable suggestions. I want to thank my colleagues in the group Saad Nasir, Samantak Gangopadhyay, Ningyuan Cao, and Insik Yoon. Dr. Keith Bowman provided me with insightful thinking ideas during my time at Internship at Qualcomm, and I did learn a lot from him.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION AND BACKGROUND

The number of Internet of Things (IoT) devices are expected to cross 26 billion by 2020. Processing such huge amounts of data will increase the load of data centers and communication links. One of the techniques to address the challenge is to enable computation and analytics near the sensor node. It is not always required to store all the sensor data. We use gesture recognition, ECG classification as some of the examples to demonstrate the proposed work on in-sensor computation. Gesture recognition is a key enabler for human machine interfaces. Human machine interfaces continue to make remarkable advances as we enable new modalities of interaction and control. Beyond the traditional keyboard and mice, such smart devices enable advanced user interfaces, like voice command and control, camera and GPS based sensors and interfaces, as well as touch screens and displays. One key requirement of such interfaces is the "always on" capability, where the sensor needs to be perpetually vigilant and look out for user commands. Enabling such a capability, typically requires prohibitively high power consumption. In particular, in camera based systems, the problem is exacerbated by the high power consumption of the pixel arrays and the interface circuits. The power cost of continuously capturing and analyzing videos is so high that most systems require physical input from the user before accepting commands. To address this issue, a "wake up" camera front-end allows a sensor node to continuously acquire videos and monitor for a trigger that will wake up the back-end when necessary, thus enabling exciting new usage models.

Fig. 1.1 (a) shows the typical gesture recognition system. It includes a pixel array followed by an analog to digital converter (ADC). Pixel array provides voltage corresponding to the intensity of the image. Analog to digital converter provides a digital code corresponding to pixel voltage. Digitized intensity values are fed to CPU/GPU to perform

gesture recognition. If the gesture of interest is found, it is quantized and transmitted to the back-end server for further processing. Fig. 1.1 (b) shows typical power numbers for different blocks involved in gesture recognition and transmission. Digital processors like CPU/GPU dominates the power of the entire system. In our work we propose sensors and data-converters which also embed the computation within itself.

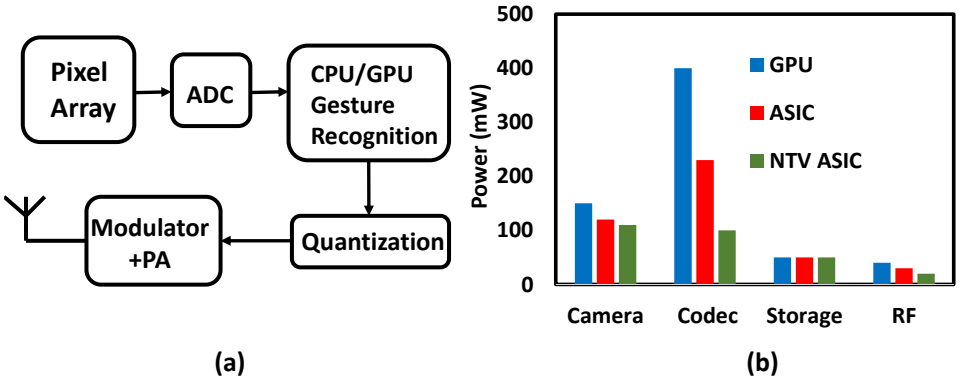


Figure 1.1: (a) Typical gesture recognition system (b) Computing power across different components [1]

Fig. 1.2 illustrates the landscape of self-powered sensor nodes and shows the power requirement of various electronic devices and the amount of power that can be harvested from various sources like solar energy, thermal, mechanical etc. In particular, for image/video processing and classifications we need high computational power. CPU, GPU and FPGA's are typically used to perform gesture recognition and object classification on video data [2, 3]. However, for "always on" front-ends where the objective is trigger identification and not continuous gesture recognition, high performance (and hence high power) are not optimal. Instead vision specific MCUs, DSPs and ASIC's are more attractive for self-powered devices, since they exhibit: (1) power dissipation in the order of hundreds of mWs (a 10X reduction compared to CPUs), (2) compact size and low thermal requirements, (3) sufficient computational ability for "always on" applications. We present an "always-on" system using Black-fin DSP processor. Camera front ends which require column parallel

ADC's consume power in the range of 10's of mW and ADC's in itself consume power in the range of 100μW-1mW. Therefore, it is crucial to reduce the ADC power. We propose voltage and time based matrix multiplying ADC (MMADC) designs. We demonstrate circuit topologies as well as measurement results. MMADCs act as an in-sensor computation element which reduces the power of the back-end processing significantly. We propose fixed point, non-linearity aware training procedure which maintains accuracy compared to floating point implementations and reduces the overall system power by an order of magnitude.

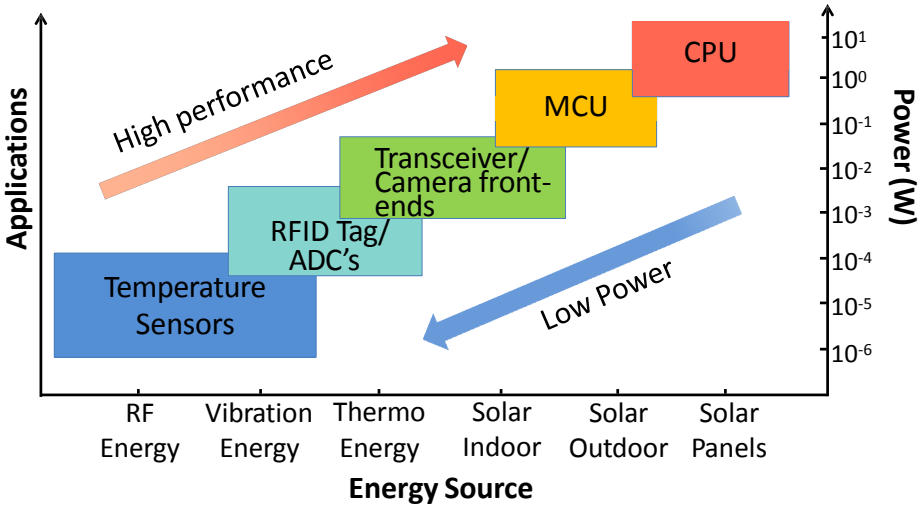


Figure 1.2: The landscape of self-powered electronic devices

Autonomous micro-robots are finding a significant application in disaster management. Robots can go into areas where to remotes areas to perform critical works like turning off the valves which are considered to be dangerous to be carried out by humans. One of the essential aspects of the micro-robot is the integration of sensor, actuator, the processor within itself. In most of the disaster management areas, the robot will not have access to GPS. Therefore, edge computing becomes very crucial. We propose an autonomous obstacle avoidance robot through reinforcement learning for this purpose.

CHAPTER 2
LITERATURE SURVEY

Reported works on gesture recognition, low power sensor front-end circuits are described in this section.

2.1 Gesture recognition systems

Traditional gesture recognition systems are power inefficient and run on batteries or even AC power supplies [2, 4]. A non-appearance based low-power gesture recognition system [5] has been proposed. Relying on the wireless signal trigger by the gestures, the performance of this system is limited by the short sensing distance and is sensitive to environmental noise. They recognize very simple gestures like the flick, push etc. For the security-related applications, we need to be able to recognize more complex gestures. Reported works in [2, 4] show gesture and posture classification using FPGA and CPU respectively. Both of the demonstrations require high voltage DC supplies. To have better portability, mobility and longer battery life, we envision designs where the gesture recognition system is sustained by energy harvested from the environment. We, therefore, focus on reducing the energy consumption of the more robust appearance-based approaches. With rapid advances in energy harvesting, it is enticing to think about a camera front-end which is powered by photo-voltaic cells (PV), thus paving the way for light-powered, smart, "always on" cameras.

Typical gesture recognition system includes pixel array followed by an analog to digital converter. Pixel array provides a voltage corresponding to the intensity of the image. Digitized intensity values are fed to CPU/GPU to perform gesture recognition. If the gesture of interest is found, it is quantized and transmitted to the back-end server for further processing. It is highly crucial to use ASIC's, DSP processors for computation and reduce

the power.

2.1.1 Always ON image sensors

Previously reported works [6] [7] feature "always on" image sensors to capture images. Even though they perform low-power image capture, they don't perform activity recognition. The power for computation also needs to be reduced to a significant extent. The power, energy spent in transmitting the data is not accounted for in the reported work [6] [7].

In particular, for image/video processing and classifications we need high computational power. CPU, GPU and FPGA's are typically used to perform gesture recognition and object classification on video data [2, 3].

2.2 Low power camera front-end circuits

ADC produces a digital output corresponding to pixel intensity. ADC consumes most of the power in an image sensor. There has been extensive work on compressive sampling high speed ADC architectures reported in [8] [9] [10]. In contrast to existing algorithms which work directly in the pixel domain [11], we extract image features in the compressive domain, which are linear random combinations of pixel values, followed by smashed filter based classification. Conventional compressed domain sensor front-ends include Nyquist ADCs followed by all-digital compression Fig. 2.1 [12]. This is typically followed by digital classifiers. For example, back-end support vector machines (SVMs) have been demonstrated in [12] for bio-medical applications. These all-digital techniques do not take advantage of the energy-efficiency of mixed signal (MS) design. Latency involved in A to D conversion and classification is non-trivial. Embedding signal conversion along with classification has recently gained attention with hardware demonstrations in [13] [14]. Embedded signal conversion and computation helps to reduce the power significantly since the classification is performed along with A to D conversion. Embedded signal conversion/classification increases the throughput of the entire system by a factor of 2-4X. The

circuits reported in [13][14] operate on static images and applications related to image processing. However, a significant portion of the embedded applications on video data-streams, like gesture recognition, localization of moving objects etc. require the system to operate on the difference of frames. Also, the A to D converters reported in [13][14] operate in Nyquist speed, therefore, power efficiency is drastically reduced. The work reported in [14] reduces energy per compute however it uses convolutional neural network (CNN) classifier which is computationally expensive and uses analog accumulators for storing the intermediate outputs. Accumulating the values in the analog domain reduces the dynamic range of the output and reduces the classification accuracy. Research reported in [13] and [14] are targeted towards object classification.

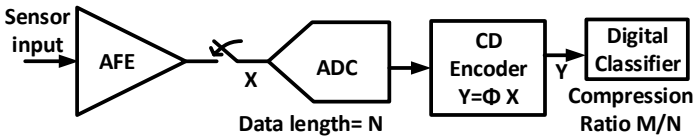


Figure 2.1: Traditional CD classifier [12]

Most of the video processing/object localization tasks operate on the difference of image frames. Object localization algorithms require parameter such as motion centers to be extracted. This is a highly computationally expensive process [15][16].

2.3 ADC Architectures for CS Image Acquisition

In prior work for obtaining compressed domain data, both analog and digital techniques have been used to perform compressive measurements from the raw data. Typically, analog implementations of compressed sensing suffer from low Signal to Noise Ratio (SNR) and require an analog to digital converter to improve accuracy[17][18]. Resistor based compressed sensing multiplexor reported in [19], suffers from static power dissipation and the number of inputs (n) is limited, making it suitable for RF receiver applications only.

To overcome some of the disadvantages of analog CS circuits, [20] has proposed compression in the digital domain. Fig. 2.2. a) shows the technique proposed in [20]. The entire analog signal is converted into the digital domain by high-speed ADCs and the CS encoder does compression in the digital domain. This is mostly suited for bio-medical applications where the number of input channels is limited. However, for CIS of a typical $256 * 256$ size, ADC would need to acquire all the samples and then convert to the digital domain. The number of measurements by the ADC will not be reduced and it defeats the purpose of compressed domain data acquisition. Further, the size of digital CS encoder grows exponentially with the number of inputs. CS encoders will further add significant power along with the ADC making it infeasible for "always on" imaging applications.

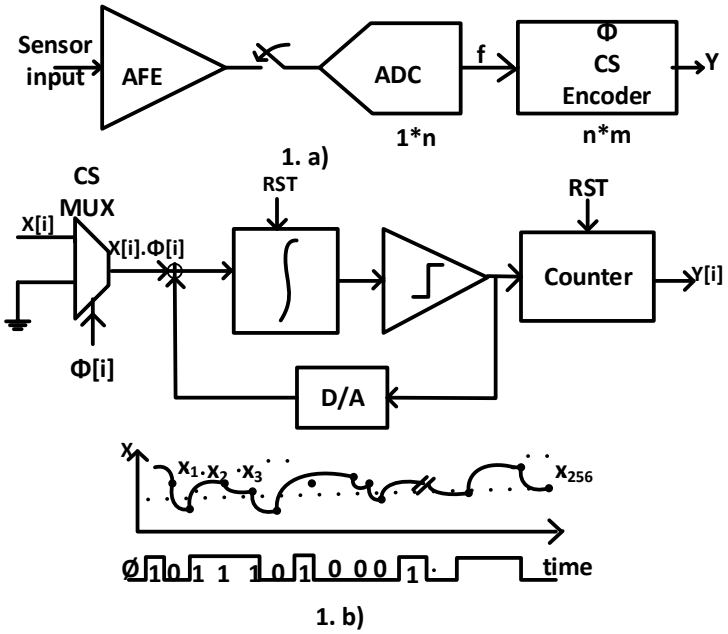


Figure 2.2: a) CS encoder on acquired samples of ADC[20] b)Simultaneous compression and quantization within [21]

Fig. 2.3 shows the smashed filter based object classification technique. Input image/signal is vectorized into a single column format. The vectorized column (D_{i1},...,D_{iN}) is multiplied using the random matrix of size M*N (where M<<N). The compressive measurements are denoted by $(y_1, ...y_n)$. The number of compressive measurements is given by: ($M * N$/No. of parallel inputs or processing). Vectorized K templates as $X(\alpha)$, where

α represents the training set elements.

$$\hat{\alpha}^* = \arg\max_{\alpha} X^T(\alpha)\Phi^T.y \tag{2.1}$$

The above process is known as smashed filtering [22] which is akin to matched filtering in the CD. It can be decomposed into multiplication with a compression matrix Φ, a smashed filter matrix Ψ (α), followed by WTA (Fig. 2.3 (b)).

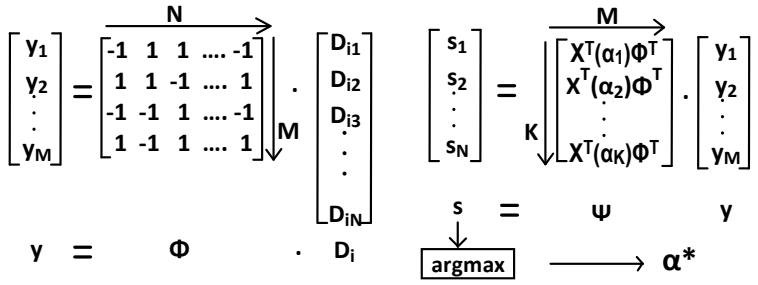


Figure 2.3: (a) Compressed Measurements (b) Classification of compressed measurements using smashed filter

To overcome the limitations of data acquisition followed by compressed domain measurements, Oike et.al, has proposed a CS camera through simultaneous averaging and quantization of pixels using a $\Sigma - \Delta$ ADC [21]. Fig. 2.2. b) shows the schematic of the resetting $\Sigma - \Delta$ ADC used for such linear measurements. Pixel values are multiplied with random numbers (from the ϕ matrix) sequentially and passed to the input of the $\Sigma - \Delta$ ADC. This approach requires m measurements; however it requires n conversion cycles for one measurement. This architecture requires a $16 * 16$ block for linear measurement. For each measurement of the block, the resetting $\Sigma - \Delta$ ADC needs 256 clock cycles. During this conversion period, all the high gain amplifiers will remain on and consume power. Hence, for lowering the total power dissipation, faster conversion with the opportunity for power gating once the conversion is complete, will be critical.

Once compressed domain data is acquired, the image is often used for online classification to detect potential trigger signals. For such in-situ classification [13] and trigger identification, 8bits of inputs are sufficient. Further, it has been shown that for most of

the machine learning applications moderate resolution (6-8bits) is sufficient[23][24]. Fig. 2.4 a) shows the Energy per conversion with respect to the Signal to Noise and Distortion ratio (SNDR) for state of the art SAR, Pipeline, VCO and $\Sigma - \Delta$ ADCs. SNDR is related to effective number of bits ($ENOB = (SNDR - 1.76)/6$) [25]. From this plot, we can observe that SAR ADC, VCO ADC's have best FOM (order of $10 - 1000 fJ/conv$) for moderate resolution (6-8 bits). Pipeline ADCs also have competitive FOM for moderate and high-speed applications. Fig. 2.4 b) shows classification accuracy vs ENOB. We can observe that classification accuracy saturates after 6-7 bits. This makes SAR, VCO based ADC's suited for machine learning applications.

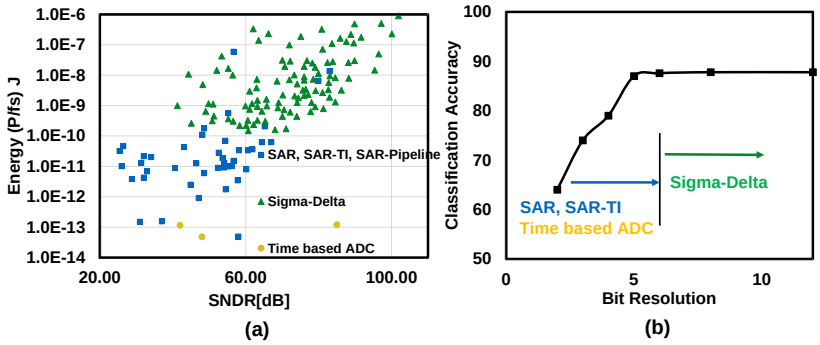


Figure 2.4: a) Energy vs SNDR for state of the art reported SAR, Pipeline and $\Sigma - \Delta$ ADCs[26] b) ADC choice with classification accuracy

Embedded sensing applications require machine learning algorithms to perform in-sensor classification [27] [28] to avoid transmission bandwidth and latency to the cloud. Low power and low supply are crucial for sensing and inference. Embedded inference using neural networks, matched filters are gaining importance [27] [14]. There has been extensive work on ADC based in-sensor classifiers reported in [13] [14] [28]. All of the published results so far, demonstrate low-precision computation in voltage domain along with data-conversion. However, the minimum supply required [13] [14] is in the range of 1-1.2V. A lower supply voltage is preferred, if we need to operate the sensors using energy harvesting sources. Also, a low supply voltage is compatible with the traditional digital back-end and is scaling friendly. Due to the scalability and energy-efficiency, VCO based

ADC's [29] [30] are gaining more attention. Fig. 2.5 (a) shows the traditional embedded sensor/classifier interface. In a traditional sensor interface, the ADC operates at a higher supply [13] and digital back end operates at low supply (0.4 - 0.6V). The traditional interface requires two different supply voltages, hence reducing power efficiency. Fig. 2.5 (b) shows the proposed embedded sensor/classifier interface. We use Adaboost with SVM as the weak classifier. We perform matrix multiplication in time domain and accumulation in digital domain. The accumulated value is compared to a bias value to obtain the results of the weak classifier. Then we perform majority voting and adaboost on the weak classifiers to obtain the final decision.

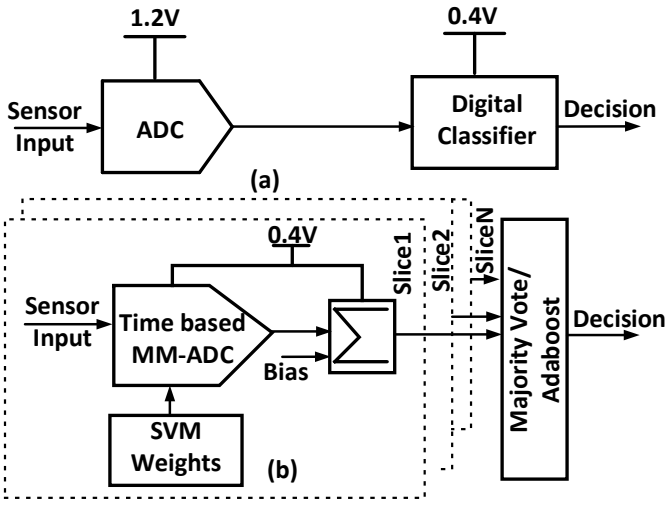


Figure 2.5: (a) Traditional sensor/classifier interface (b) Proposed sensor/classifier interface

A VCO based ADC is shown in Fig. 2.6. It consists of a Voltage to Frequency (V to F) converter and Frequency to Digital Converter (FDC). The V to F converter is achieved using a VCO. The FDC is a counter operating at the VCO frequency. Hence, the counter output is proportional to the input Voltage. With the proposed architecture, the sensor front-end can operate at a scaled digital supply. Such a digital ADC is easily portable from one technology to another and easy to interface with the digital back-end circuits. VCO based ADC is highly scalable with supply voltage and can operate till 0.4V.

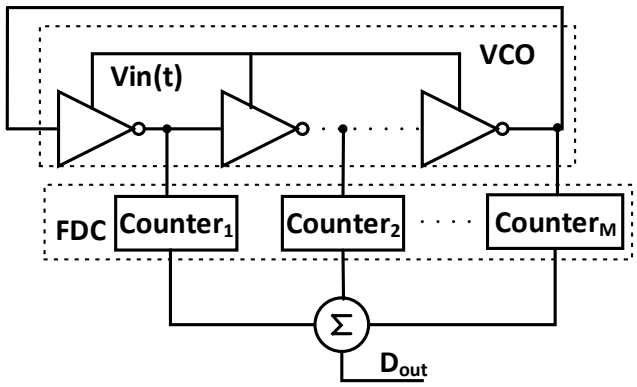


Figure 2.6: Traditional VCO based ADC [12]

CHAPTER 3

ALWAYS ON COMPRESSED SENSING GESTURE RECOGNITION

We brief the hardware system architecture. The proposed system consists of four main components: a PV cell array, a DC-DC converter with output voltage regulation, an MCU, and an image sensor. The block diagram of our system is shown in Fig. 3.1. Camera and MCU are powered by PV cell. If we find gesture of interest then we transmit the captured image/video. The PV cell converts solar energy to electrical energy. The Norton equivalent output current (Fig. 3.2) of PV cell is given by:

$$I = I_{pv} - I_0[exp(\frac{V + IR_s}{aN_sV_T}) - 1] - \frac{V + IR_s}{R_{sh}} \tag{3.1}$$

where I and V are PV cell's output current and voltage respectively; R_s and R_{sh} are the series and shunt resistances; I_0, V_T, a, N_s are dark saturation current, thermal voltage, diode ideality factor, and number of cell connected in series respectively; I_{pv} is the generated current whose magnitude depends on irradiation and temperature.

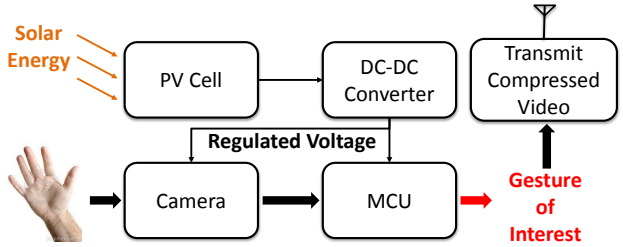


Figure 3.1: Block diagram of the proposed camera front-end

As the MCU and image sensors both demand regulated voltage to operate properly, the DC voltage generated by PV cells must be regulated by a DC-DC converter. Fig. 3.3 shows the proposed light-powered, "always on" gesture recognition system. In the current design, we select TI's BQ25570EVM, a two-stage DC-DC converter with Maximum Power Point

Tracking (MPPT) for solar energy harvesting and for providing a regulated output supply. The block diagram of the energy harvesting system and gesture recognition flow is shown in Fig. 3.3. The input image is captured by Omnivision's OV7672 sensor with a native resolution of 480×640. We extract only the gray-scale component of the image, which reduces the computation power without any impact on performance. The output of the pixel array is passed on to an on-board ADSP BF707 MCU using I2C interface. Once the image is received by the BF707 processor, we perform the following operations: block averaging, frame difference, random linear measurements, motion centers extraction in compressed domain followed by gesture recognition. For block compression we extract every one out of 16 pixel values in each row and column. Therefore the block compression factor is 256

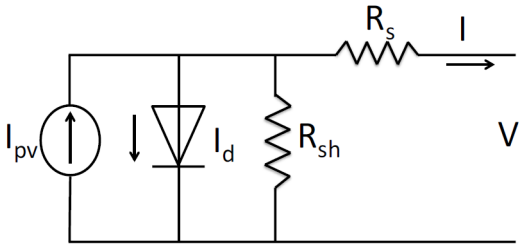


Figure 3.2: Equivalent circuit representation of a PV cell

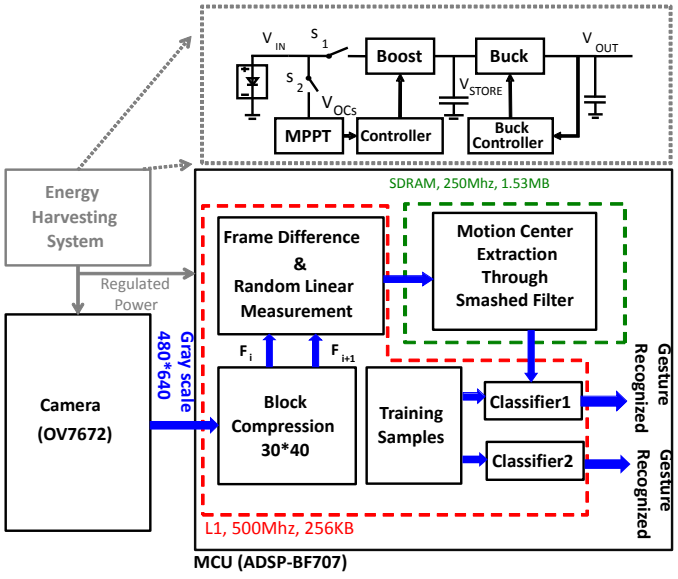


Figure 3.3: Hardware system architecture demonstrating key components of power management, the MCU and the mapping of the algorithm on the MCU.

(16 for every row and column). The block average, frame difference and dynamic time warping related matrices are stored in L1 cache (requires less than 128KB). Motion center extraction co-efficients are stored in external SDRAM (requires more than 1.2MB). L1 access is performed using core clock at 500MHz and SDRAM access happens at system clock with 250MHz speed. The hardware is further optimized by (1) using short integer maths and (2) optimizing memory usage that reduces total power consumption without loss of performance. Two classifiers are used for gesture recognition once motion center is extracted. Classifier I is the traditional K nearest neighbor (K-NN) classifier using dynamic time warping (DTW) distance measurements. Classifier II is a modification of classifier one, allowing it to cooperate with clustering and dimension reduction techniques [16]. Classifier II reduces the memory requirement in the L1 cache as well as improves the computational efficiency.

3.1 ALGORITHMS

Difference images are capable of capturing gestures containing significant motions. We pass each difference image through two layers of compression. In the first layer, the resolution is reduced by dividing the whole image into several blocks and taking the average of each block. In the second layer, we take coded combinations of these block-averaged pixels. We estimate the center of the motion directly from these compressed measurements. These motion centers are passed to a classifier for gesture recognition. Fig. 3.4 shows the block diagram of our system.

3.1.1 Two layers of compression

Denote F_i as the ith full resolution image output from the camera of size $W \times H$. The difference image D_i (Fig. 3.5. (a) of two consecutive frames is calculated as $D_i = |F_{i+1} - F_i|$.

In the first compression layer, the difference image is divided evenly into blocks of

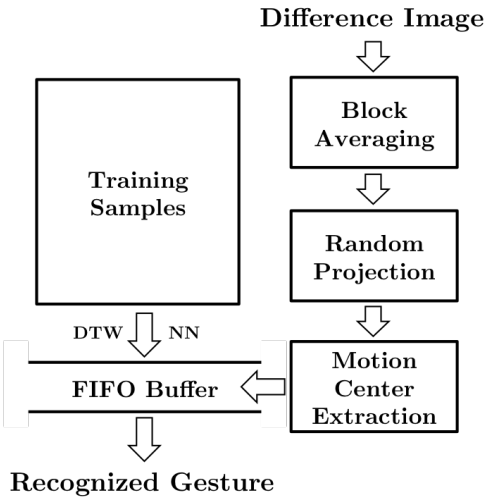


Figure 3.4: Block diagram of the proposed algorithm

size B by B. The average of the pixel values in each block is taken, resulting in a block-compressed difference image of size W/B by H/B (Fig. 3.5 (b). We vectorize this low-resolution difference image and denote it as $y_i \in R^N$.

In the second layer of compression, we chose a random matrix Φ of size M by N as the coded measuring matrix. Each entry of Φ is uniformly chosen from $\{+1, -1\}$. The projection of the vectorized low-resolution difference image in the compressed domain is calculated as:

$$\hat{y}_i = \Phi y_i = \Phi \Psi Y_i \tag{3.2}$$

Each entry in $\hat{y}_i \in R^M$ is a random linear combination of all the entries in y_i. Y_i is the vectorized original difference image D_i. Ψ is the block averaging matrix of size N by $W \times H$.

3.1.2 Motion center extraction in the compressed domain

In the uncompressed low-resolution domain, the hand region in the difference image can be captured by a template shown in Fig. 3.5 (c). The template (of size W/B by H/B) has uniform non-zero values within the small rectangular region and zeros elsewhere. To locate the hand region, we construct a set of vectorized templates $X(\alpha, r)$, where α represents the

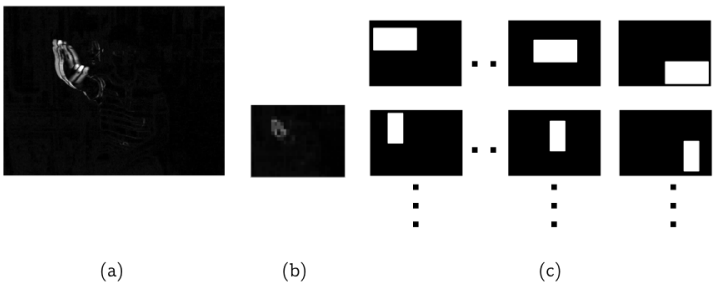


Figure 3.5: (a) Full-resolution difference image D_i; (b) Block averaged difference image; (c) Matching templates in the uncompressed domain. Rectangle sizes differ among rows and centers of the rectangles differ among columns.

coordinates of the center of the small rectangle, and r represents different rectangle sizes. The variation in sizes is to adapt to the change of the hand size seen by the camera when users at different locations. The center of the hand motion is extracted by solving

$$(\alpha^*, r^*) = \arg\min_{\alpha,r} ||y_i - X(\alpha, r)||_2 \tag{3.3}$$

The collection of templates forms a manifold in R^N with intrinsic parameters α and r. Using the result from [22] and [31], we can directly extract the motion centers in the compressed domain. That is, for

$$(\hat{\alpha}^*, \hat{r}^*) = \arg\min_{\alpha,r} ||\hat{y}_i - \Phi X(\alpha, r)||_2 \tag{3.4}$$

$(\hat{\alpha}^*, \hat{r}^*) \approx (\alpha^*, r^*)$ with high probability for some $M \ll N$. The block averaging layer reduces the possible choices of r, and techniques such as matched filtering can be applied to efficiently solve equation (3.4).

3.1.3 Train the gesture classifier II

DTW-based classifiers perform well for dataset containing limited amount of samples [32]. Traditional DTW-based classifiers use DTW [33] as the distance measuring method between two sequences of different lengths, and use K-NN method for classification. The memory and computational requirements thus grow linearly with the size of training set.

To reduce the number of DTW calculations in the recognition stage, we perform K-

means clustering in the training dataset to form "super samples". The distance between an individual sample and a super sample is measured using DTW. In each iteration, the super samples are updated as the average of all the samples within their clusters. DTW barycenter averaging (DBA) [34] is used as the averaging method.

The main difficulty of time series classification comes from the different lengths of the samples. We notice that DBA can find clustering centers of an arbitrary length set by the user. The pairwise matching information in DTW also provides a way to rescale the length of time sequences. Therefore, we propose a DTW length rescale algorithm, shown in Algorithm 3.1.3: [H] K "super samples" $\mathbf{S_1}, \mathbf{S_2}, \ldots, \mathbf{S_K}$ of length τ, calculated using K-means with DTW and DBA. Sequence $\mathbf{T}$ to be rescaled to length τ $\mathbf{S}^* = \arg\min DTW(\mathbf{S_k}, \mathbf{T})$ $M \leftarrow$ pairwise matching information between $\mathbf{S}^*$ and $\mathbf{T}$ Initialize $\mathbf{T'}$ of length τ $i = 1$ to τ S_i^* is matched to multiple points $T_j, T_{j+1}, \ldots, T_{j+m}$, according to M, $T_i' = \underset{T_l \in T_j, \ldots, T_{j+m}}{\arg\min} ||S_i - T_l||$ $T_i' = T_j$, where T_j is the only matching point to S_i^*

Using Algorithm 3.1.3, we rescale all the training sequences to the same length τ. Since each motion center in the time sequence contains both x and y coordinates, each gesture sample is now of size $2 \times \tau$. We vectorize the samples by cascading all the y coordinates after the x coordinates, transferring the time series classification problem to a traditional classification problem in $R^{2\tau}$. Various dimension reduction techniques and multi-class classification algorithms can then be implemented. The block diagram of the complete training procedure is shown in Fig. 3.6.

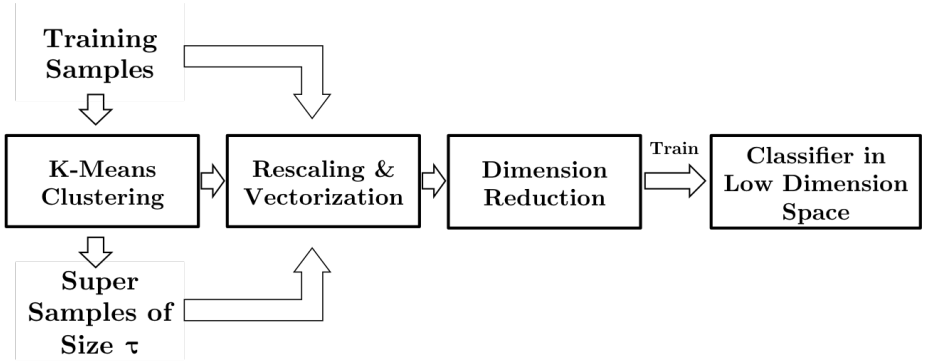


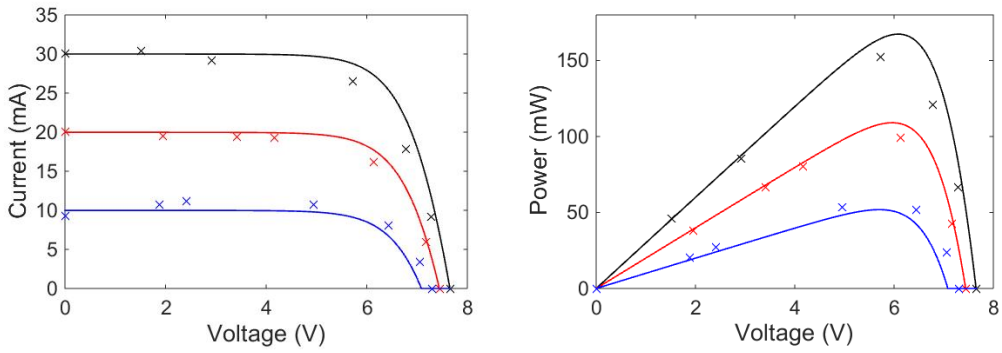
Figure 3.6: Block diagram of training the gesture classifier

3.2 Experimental setup

3.2.1 Power Management Design

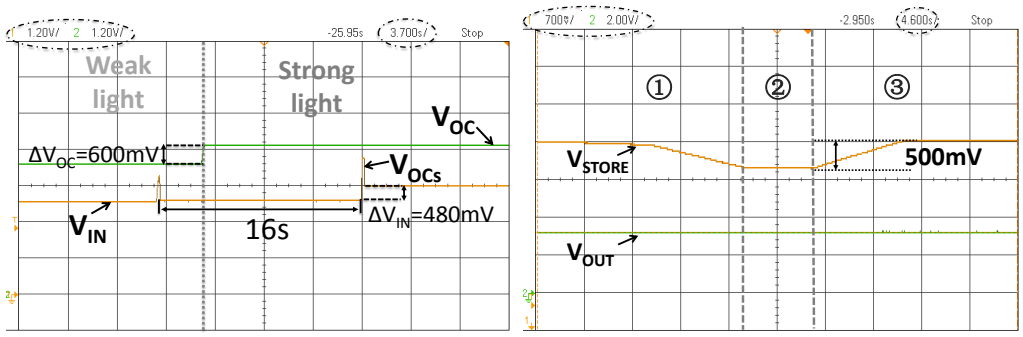
The overall platform is designed from COTS components and here we explain the optimal design choice. We chose omnivision OV7672 image sensor which has frame size of 480×640 pixels. The image sensor is connected to an ADSP BF707 processor using I2C interface. Measurements reveal a maximum current consumption of $170mA$ at fixed $3.3V$ power supply.

The solar cell (AM5907) produces an output voltage of 5V at the point of maximum power transfer. The I-V and P-V characteristics of each cell is shown in Fig. 3.7a and Fig. 3.7b vis-a-vis simulation results. We see a close match between experimental results and empirically fitted Eq 3.1. We note that for an irradiance of $600W/m^2$, the maximum power $\approx 100mW$. In the current setup, We use 6 PV cells in parallel to generate the required power that the load demands. Also, from Fig. 3.7b, we observe that operating voltage at maximum power point is approximately 80% of the open circuit voltage (V_{OC}). Hence, Maximum Power Point Tracking (MPPT) is achieved by regulating the output at 80% of V_{OC}.



(a) I-V characteristics (b) P-V characteristics

Figure 3.7: (a) I-V characteristics and (b) Power-Voltage characteristics of PV cells at three different irradiance levels (300W/m^2, 600W/m^2 and 1000W/m^2). Discrete points are experimental results and continuous curves are from simulations.



(a) (b)

Figure 3.8: Oscilloscope captures illustrating (a) MPPT where V_{IN} tracks the open circuit voltage at 80% of V_{OC} as irradiance changes and (b) regulation of V_{OUT} under dynamically varying super-capacitor voltage (V_{STORE}). In (b) three regions are shown: (1) instantaneous load power consumption is higher than input power which reduces V_{STORE}; (2) load power and the harvested power are balanced and (3) load consumption is less than harvested power.

Figure 3.3 shows, the MPPT block samples open circuit voltage V_{OC} every 16 seconds with S_2 on and S_1 off. This sample voltage V_{OC_sample} is sent to the boost controller to modulate the phase and frequency of the boost converter so that the PV cell operates at maximum power point, 80% of V_{OC_sample}. The sampling process is shown in oscilloscope captures in Figure 3.8a. It is observed that open circuit voltage is sampled and the PV cell's operating voltage changes accordingly. The energy is stored in a super-capacitor between the two converter stages. Availability of super-capacitor benefits camera-based applications whose power requirement fluctuates significantly. The output voltage is sensed and sent back to buck controller to regulate the output voltage. The output voltage is hardware programmable through programmable external resistors on the board. Fig. 3.8b shows how V_{STORE} varies with varying irradiance and load current conditions. Measured oscilloscope capture also reveals that V_{OUT} is well regulated under such dynamic conditions. The complete experimental setup along with the PV cells and the MCU is shown in Figure 3.9.

3.3 Measurement results

3.3.1 Number of compressed measurements vs. Recognition Rate and Power Consumption

For different numbers of compressed measurements, we measure the energy consumption per frame and the recognition rate. We evaluate 20 gestures of each class, and the recognition rate is calculated from the total number of correctly recognized gestures. The total time per gesture is kept at 0.1secs. In a typical instance, the motion centers for a gesture "Z" as extracted from the hardware is shown in Fig. 3.11 (a). We have also plotted the measured motion centers for N gesture in Fig. 3.11. (b). Fig. 3.12 a) shows the measured design space exploration. We have measured the recognition accuracy as a function of M which reveals an accuracy rate of $> 80\%$ for $M > 300$, which closely matches simulation results described in the previous section. Fig. 3.12 b) shows dependence of the power consumed by the MCU and the corresponding recognition accuracy of the proposed system as a function of the frame rate. We note that a minimum frame rate of 5fps is required for maintaining a desired recognition accuracy of [$\geq$84%]. As the frame rate increases, the corresponding power consumption also increases and shows a graceful trade-off between accuracy and power consumed.

As the environmental conditions and irradiance levels change, the proposed system can scale the frame/sec accordingly, which gracefully trades-off recognition accuracy. Fig. 3.13 illustrates the trade off between recognition accuracy for different gestures as a function of

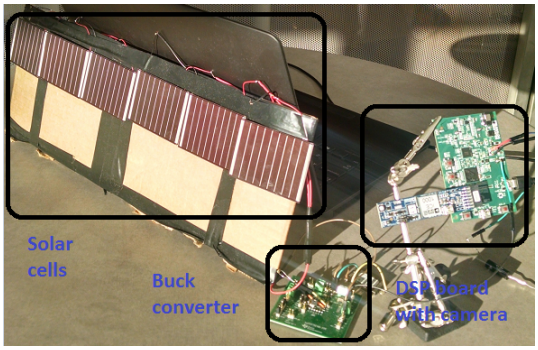


Figure 3.9: The overall system demonstrating the solar cells and the MCU with the camera.

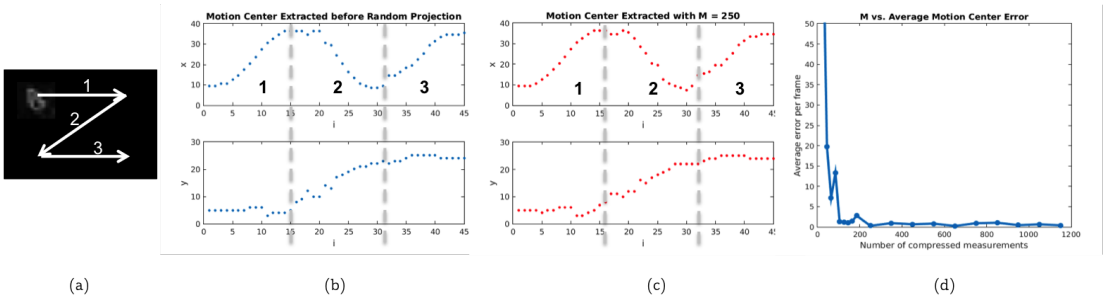


Figure 3.10: Simulation results (from MATLAB) for different numbers of compressed measurements M. (a) Hand motion of gesture "Z" divided into three segments, (b) Motion center extracted before random projection by solving equation (3.3), (c) Motion center extracted from 250 compressed measurements by solving equation (3.4), (d) Average error of motion center extracted in the compressed domain compared with (b).

Irradiance.

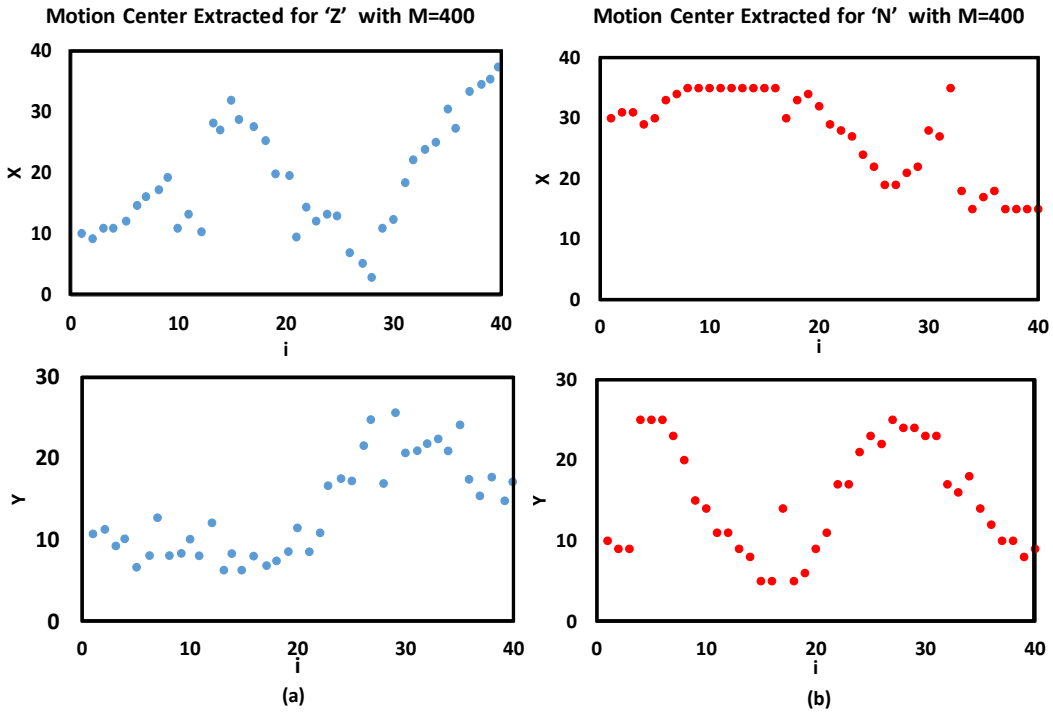


Figure 3.11: a) Measured motion center extraction of hand gesture "Z" for $M = 400$. b) Measured motion center extraction of hand gesture "N" for $M = 400$

3.2 System's Multi-class Recognition Accuracy

At $M = 400$ the recognition accuracy of 5 different gesture classes are shown in Table 3.2. It can be seen that the recognition accuracy depends on complexity of the gesture. For a simple gesture, e.g., "+", a peak accuracy of 90% in a fully solar energy harvested system is measured. A comparison of the proposed system with competing hardware [4,

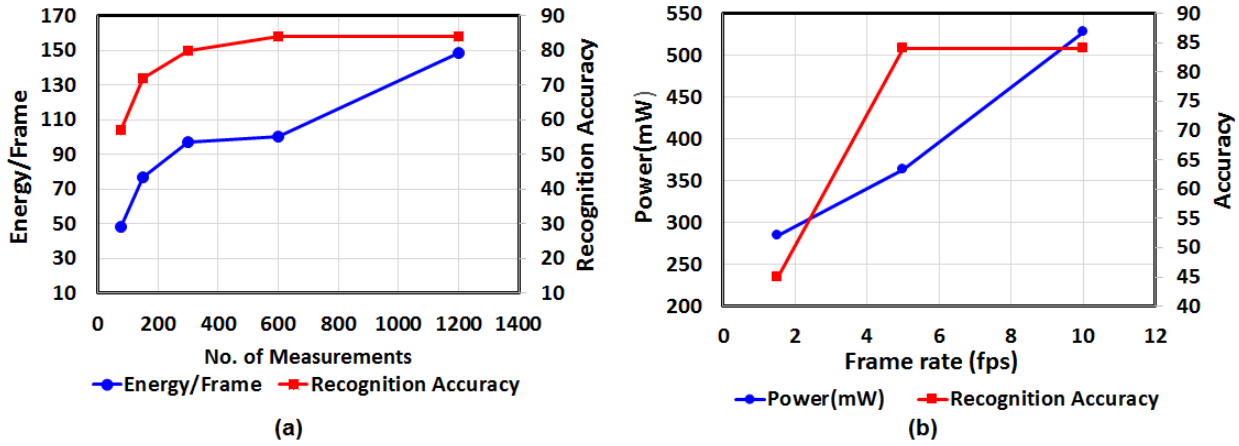


Figure 3.12: Design space exploration through measurements: (a) Number of compressed measurements (M) vs. Energy/frame and Recognition accuracy (b) Frame rate vs. Power and Recognition accuracy

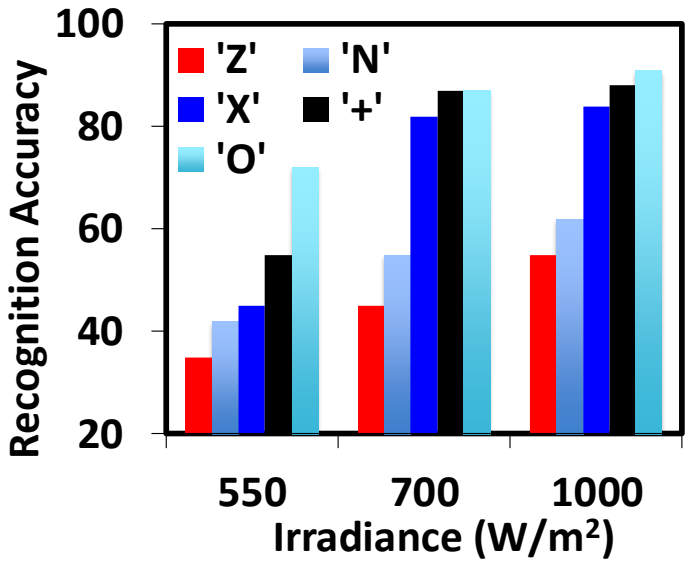


Figure 3.13: Recognition accuracy vs Irradiance for Z, X, O, N and +

3, 35] based motion and gesture detection is shown in Table. II. The proposed system demonstrates more than $3\times$ improvement compared to reported works in energy/frame for detecting "wake up" gestures. This enables a fully self-powered "always on" camera front end.

Table 3.1: Recognition accuracy of 5 gesture classes (M=400)

Gesture Type	+	Z	X	O	N
Recognition Rate	90%	70%	85%	85%	75%

Table 3.2: Comparison with state of the art gesture recognition systems

Gesture Type	Daeho[4]	Chao Li[3]	Yu Shi[35]	Chen[36]	Proposed system
Detection and Tracking	Motion and K-cos features	Optical flow Kalman filter	Chain mode row representation	3d motion and SVM	Motion center, DTW
System Processor	Pentium PC 2.5GHz core 2DUO	FPGA EPC270	FPGA 2VP30	Xbox 360	ABI BLIP2 BF707
Frame rate	30fps	30fps	30fps	30/60fps	10fps
Energy/frame Compute	0.5J	0.366J	0.366J	16μJ	95mJ
Application	Motion detection	Flick gestures	Posture	Hand gesture	Gesture $+, Z, X$
Power	240V AC	240V AC	240V AC	240V AC	Solar

CHAPTER 4

VOLTAGE AND TIME BASED COMPRESSIVE SENSING ADC

4.1 Compressive sensing for image classification

Recently developed algorithms of compressive sensing (CS) promise to reduce the number of measurements with non-linear recovery at the back-end [37]. If the pixel values in a camera are represented as a discrete time signal $X = [x_1 x_2 x_3 x_4 \cdots x_n]^T$, the number of measurements needed in traditional column parallel ADCs will be equal to n. Instead of n samples, CS needs only m linear measurements ($m << n$). The CS measurement matrix is given by Eq. 4.1.

$$Y[m] = \phi[m,n] \times X[n] = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & \cdots & 0_{m,n} \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{4.1}$$

Here $Y[m]$ is the m-dimensional measured array, ϕ is a random binary matrix of size $m * n$ and follows the "Independent and Identically Distributed (IID)" property. X is traditionally recovered at the back-end using an optimization algorithm, like determining the L_1 norm[37].

In this work we present a novel pipeline-SAR ADC architecture with capacitive DAC sharing with the capability of acquiring linear combinations of 64 pixel data in a single conversion cycle. This is suitable for such compressed domain data acquisition.

4.2 SAR-Pipeline ADC Architecture for CS Measurements

For most of the low power applications SAR ADCs are used since they consume ultra-low energy per conversion(Fig. 2.4). But they are limited to low sampling rate & low resolution because of the DACs settling time for conversion (mainly set by MSB transition)[38] and large capacitance (for higher resolution). To alleviate this problem two-stage SAR Pipeline ADCs are proposed[38][39]. SAR-Pipeline uses two stage SAR-ADC and an amplifier which is used for amplifying residue generated by stage 1 SAR ADC (Fig. 4.1). Both the SAR ADC stages operates in parallel and each stage has to resolve lesser number of bits (lesser DAC settling time & capacitance as compared to traditional SAR). Therefore, SAR-Pipeline ADCs can operate at much higher speeds with high area efficiency [39]. One of the inherent advantage of SAR-Pipeline is residue voltage of Stag-1 SAR ADC is generated within its DAC after conversion phase. Hence this avoids extra DAC and clock phase to generate residue of Stage-1 unlike in traditional flash based Pipe-lined ADCs[39].

Fig. 4.1 illustrates the proposed ADC architecture. The design operates on a block size of $16 * 16$ (256 elements in pixel array). We propose SAR-Pipeline ADC consisting of 64-inputs Stage-1 SAR ADC resolving 4 bits (with 1bit redundancy) and Stage-2 SAR ADC resolving 5 bits. We propose time-interleaved DAC sharing for Stage-1 SAR ADC which provides a linear measurement of 64-inputs in a single cycle. We also share the amplifier (used for residue amplification) between 2 neighboring column parallel ADCs to save power. 64 inputs are simultaneously averaged and quantized using the SAR-Pipeline ADC. Post-conversion, 4 consecutive samples are averaged using a 10 bit accumulator and shift register. This allows us to average 256 samples in 4 sampling cycles.

Fig. 4.2 shows a previously reported multi-input SAR ADC used for compressed sensing (with 8 bit resolution). It uses charge sharing. The MSB capacitor is equally divided among the inputs. Because of charge sharing the inputs will get averaged after the sampling cycle. [40] demonstrates a 4 input CS SAR ADC for wireless applications. This

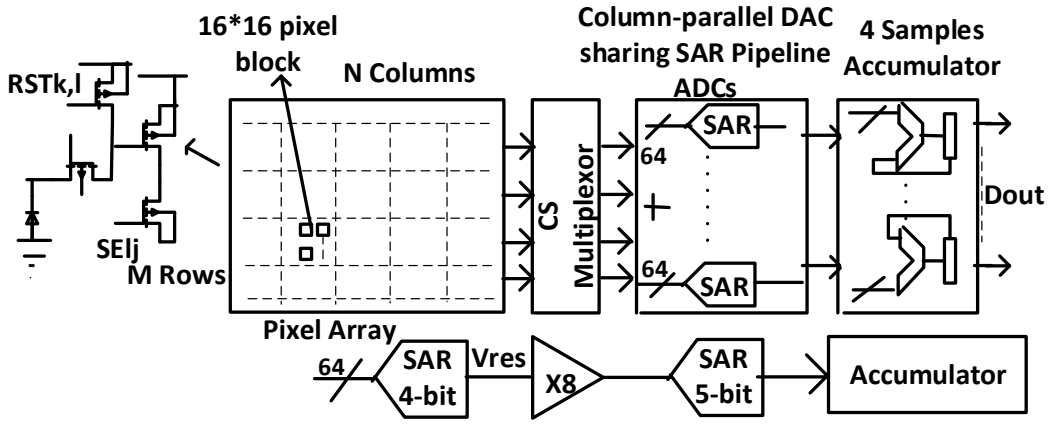


Figure 4.1: Proposed CS front-end architecture for CIS

technique requires $(2^8 + 2^4 = 272C)$ number of capacitors for an 8 bit ADC and measures 256 inputs (C is the unit capacitor). One of the main limitations of the proposed SAR ADC architecture for portable application is the area occupied by the sampling capacitors[41]. Dividing the MSB capacitors to accommodate 256 inputs requires 256 switches. For portable applications limited supply $\approx 1 - 1.3V$ provides high R_{ON}. This provides us the time constant (τ_{conv}) for conversion (min. sized capacitor of 50fF) of $\approx$ 220nsecs (DAC settling time). This allows a maximum sampling frequency of 730KHz. Hence, for high speed cameras (with 30frames/sec) the proposed ADC architecture will not be able to meet latency requirements.

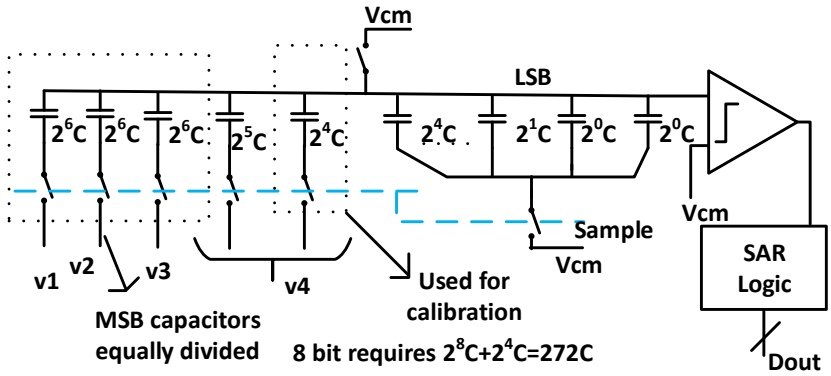


Figure 4.2: Reported multi-input SAR-ADC[40]

Fig. 4.3 is the proposed SAR-Pipeline with DAC sharing. We use 4 bit ADC as the first stage. Since 4-bit ADC has 16C capacitors, all the capacitors are divided into equal

value of C and 16 inputs are applied. We have 3 instances of the same DAC which is used for accessing additional 48 inputs. Sampling is done in two phases. During sampling phase (S1) all 4 DAC's sample 16 inputs each. During second phase of sampling charge is redistributed between them. The averaged voltages across 4 DAC's during S1 phase given by Eq. 4.2.

$$\begin{aligned} V_{dac1} &= \frac{(v_1 + v_2 + ... + v_{16})}{16} \\ &\vdots \\ V_{dac4} &= \frac{(v_{48} + v_2 + ... + v_{64})}{16} \end{aligned} \tag{4.2}$$

During the second sampling phase S2, averaging of V_{dac1} to V_{dac4} takes place. Therefore, the final voltage across DAC is given by Eq. 4.3.

$$\begin{aligned} V_{dacf} &= \frac{(V_{dac1} + ... + V_{dac4})}{4} \\ V_{dacf} &= \frac{(v_1 + ... + v_{64})}{64} \\ V_{dacf} &= \frac{(X[0].\phi[0] + ... + X[63].\phi[63])}{64} \end{aligned} \tag{4.3}$$

We can observe form Eq. 4.3 that the final accumulated output represents the dot-product of the input pixel vector X with the sampling matrix, ϕ . ϕ can be random or programmed so that both random as well as structured compressed measurements can be obtained. As soon as S2 is done 3 DAC's are shared with neighboring column parallel ADC. Once the conversion in 4-bit SAR ADC is complete, we amplify the residue by 4x and pass it to a 5-bit fine ADC to resolve the LSBs. Ideally a gain of 16 is required for residue amplification. We use 1-bit digital redundancy in Stage 1 and half reference scaling for Stage 2 to reduce the gain requirement which helps to reduce the power in the high-gain op-amp [38].

Since all the capacitors we use are identical and of value C, calibration is not required (more details in section III). As 3 DAC's are shared with 4 ADC's, we need an additional capacitance of 12C. With 12C extra capacitance we can acquire linear measurements of

64 inputs in each conversion cycle. This DAC shared method significantly improves area efficiency and enables simultaneous acquisition of multiple inputs. In this architecture, the conversion time-constant (τ_{conv}) is determined by the 4-bit ADC settling time even tough we are sampling 64-inputs. This makes the architecture suitable for high speed sensing with large number of inputs.

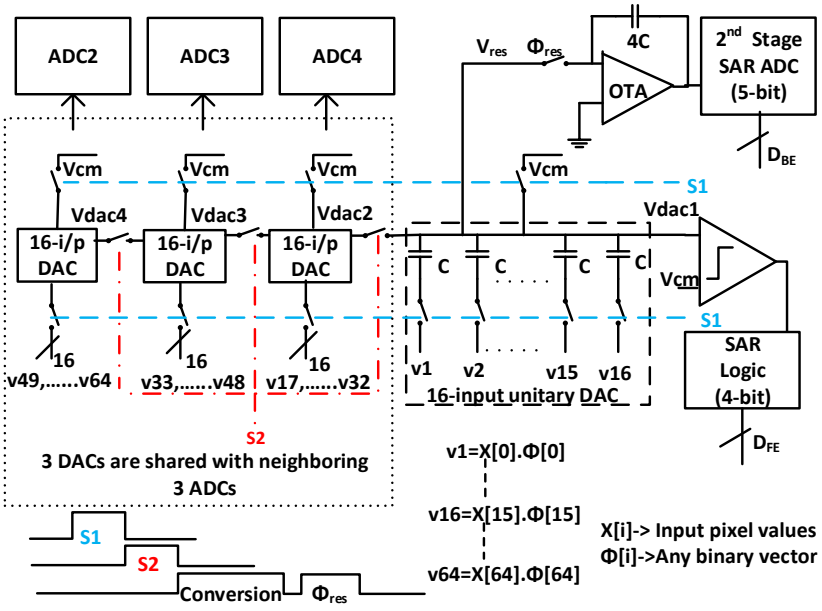


Figure 4.3: Proposed multi-input DAC sharing SAR ADC

Fig. 4.4 shows how the sampling schemes are time-interleaved for the entire column parallel ADC architecture. Conversion cycle for ADC is 8 clock cycles. During this period we share 3 DACs with 3 of the neighboring ADCs. S3 to S8 are sampling phases of ADC2 to ADC4. S3 to S8 phase operates during conversion period of ADC1. Pipelining facilitates overlapping of Stage-1 and Stage-2 conversion phases. 1-bit redundancy is added in the first stage to accommodate capacitor mismatch and offsets of the comparator, amplifiers[38]. We also share residue amplifier between two neighboring ADCs to reduce the total power[39]. Accumulator (10 bit) used to average 4 consecutive ADC output samples. The accumulator is reset after every 4 sampling cycles (F_s). The sampler operating at quarter sampling rate is used to capture the averaged output. Fig. 4.4 also shows the control logic used for proposed CS front-end ADC architecture. Global reset (RST) is used gener-

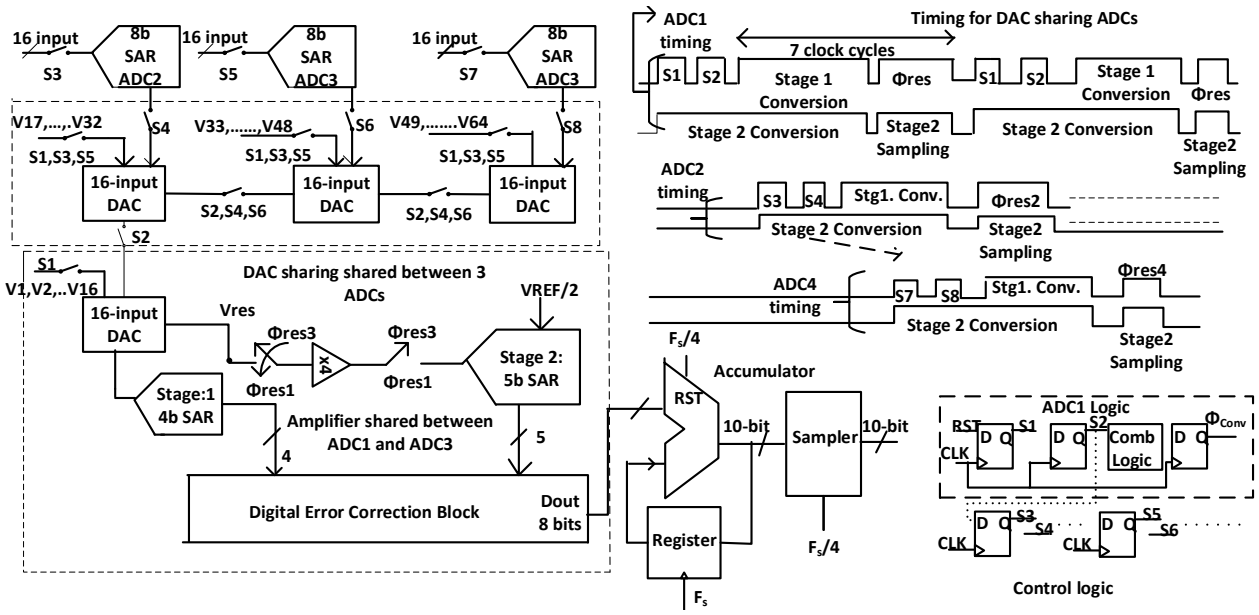


Figure 4.4: Proposed SAR ADC with time-interleaved DAC sharing

ate S1, S2 and conversion phase for ADC1. S2 phase of ADC1 is used to trigger sampling phase for neighboring column parallel ADC. This process is continued for all 4 ADCs.

4.3 Design Components

In this section, the design details of the first and the second state of the ADC are discussed.

4.3.1 Stage 1 ADC and residue amplification

Fig. 4.5 shows the Stage 1 of the proposed SAR-Pipeline ADC. 64 inputs are acquired from S1 and S2. Residue is fed into an amplifier with gain of 4. Stage 1 of the ADC has 4 bit resolution with 1 bit digital redundancy. 1 bit redundancy is used to accommodate the residual offset of the comparator, op-amp and capacitor mismatch errors.

The Op-amp open loop gain (A_{OL}), unity gain frequency (f_u) and swing ($Vp-p$) target based on the inter-stage gain is given in Table. 4.1. The Target values are derived as per gain error, gain bandwidth (GBW) requirement of the OTA to be within 1/2LSB of the ADC error[39]. Simulated values across process corners are much above the Target values.

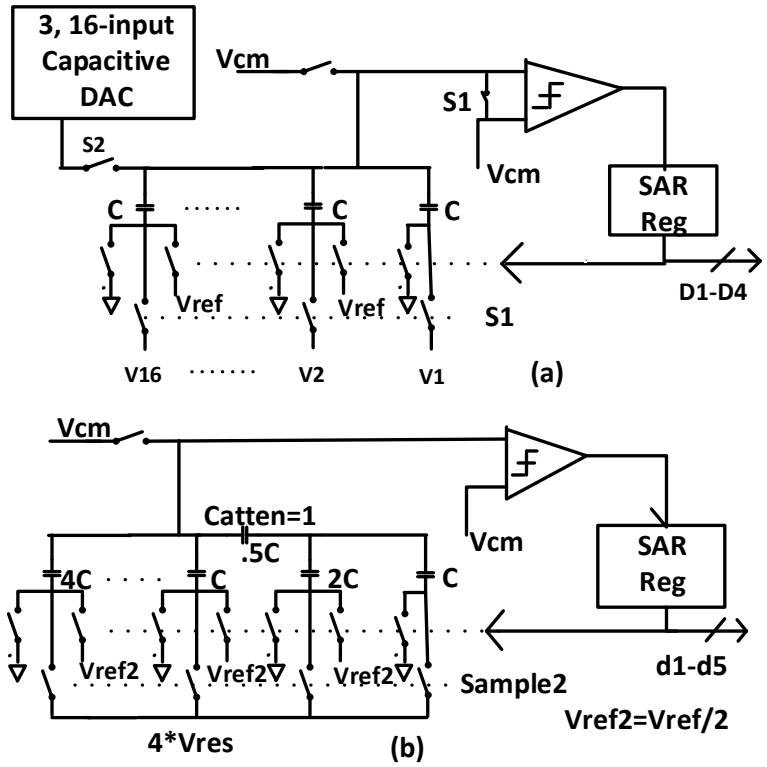


Figure 4.5: Stage 1 and Stage 2 of the proposed ADC

Table 4.1: Design requirement for amplifier and 2nd Stage offset

Inter-stage gain	Op-amp			2nd stage SAR
	A_{OL}	f_u	$Vp-p$	Offset
Target	42dB	42MHz	250mV	16.125mV
Simulated values	50dB	80MHz	300mV	8mV

We use pre-amplifier with output offset compensation to limit the offset of Stage 1 SAR ADC. The residual offset ($V_{os,res}$) is given by Eq. 4.4.

$$V_{os,res} = \frac{V_{os,pre-amp}}{A_p} + \frac{V_{os,latch}}{A_p} \tag{4.4}$$

where $V_{os,pre-amp}$ and $V_{os,latch}$ are the pre-amplifier offset and latch offset respectively. A_p is the pre-amplifier gain. The 3σ $V_{os,pre-amp}$ and $V_{os,latch}$ are 5mV and 30mV respectively. The gain amplifier features a cross coupled load which provides a high gain of 15. The residual offset is 2.33mV which is 0.25LSB of the sub-ADC.

We use telescopic cascoded OTA in the proposed design for residue amplification. Tele-

scopic cascoded OTA has high power efficiency for a given gain bandwidth (GBW)[25]. Because of half gain and half reference implementation of the ADC, the open loop gain of the OTA is reduced. The OTA achieves a swing of 300mV_{p-p}.

4.3.2 Stage 2 ADC

Fig. 4.6 shows the Stage 2 of the proposed SAR-Pipeline ADC. We use a split capacitor architecture to reduce the area and power for the second ADC. Since the non-linearity of this ADC will get divided by the gain of the amplifier, it can be neglected. For the comparator in stage 2 of the proposed ADC, a pre-amplifier with gain of 3 is used since the offset requirement from it is 15mV. The total capacitance from the second ADC is 11C.

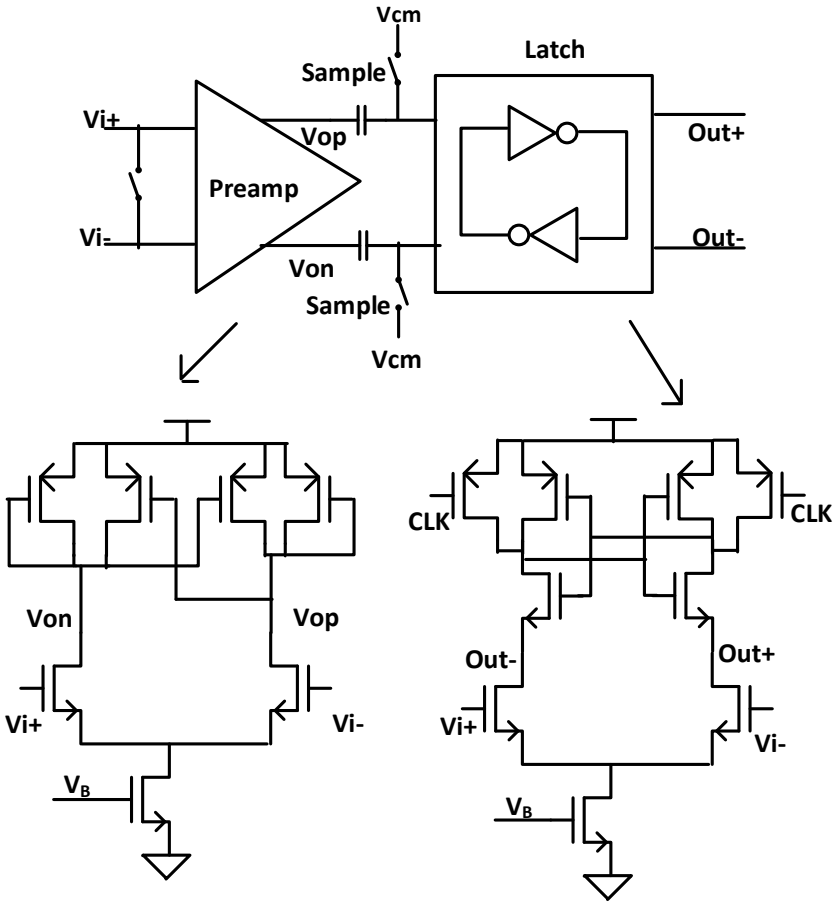


Figure 4.6: 6- bit split cap SAR-ADC for Stage 2

4.4 Analysis of Capacitor Mismatch

Systematic variations has no effect of capacitor matching since all the capacitance in Stage 1 SAR ADC are equal to C. The capacitance mismatch standard deviation for metal-insulator-metal (MiM) is given by Eq. 4.5.

$$\sigma_{\Delta C/C} = \frac{A_{\Delta C/C}}{\sqrt{WL}} \tag{4.5}$$

where $A_{\Delta C/C}$ is process constant which is $1\%.\mu$ m for 0.13μm CMOS process[42]. W &/ L are width and length of the capacitor. The minimum size allowed in 0.13μm is $5\mu m * 5\mu m$. With minimum sized capacitor $\sigma_{\Delta C/C}$ obtained will be 0.002.

As per [43] maximum allowable capacitor mismatch for a resolution of n is given by Eq. 4.6.

$$\frac{\Delta C}{C}_{max} = \frac{2^n}{2^{2n} - 2^n + 1} \tag{4.6}$$

For n=9, $\Delta C/C_{max}$ reaches close to 0.002. This shows the residue generated by first ADC will fall within the range of $1/8$LSB of error. Hence the proposed architecture is robust towards capacitor mismatch.

4.5 Simulation Results

Performance of the proposed SAR-Pipelined ADC is verified through design and simulations in the IBM 0.13μm CMOS.

Fig. 4.7 shows the normalized output frequency spectrum of the proposed ADC for input frequency (F_{in}) of 248.34kHz at sampling rate (Fs) of 1MSPS. A 1024-point FFT shows SNDR of 49.5dB which is equivalent to an ENOB of 7.9 for full scale input (dBFS). Fig. 4.8 shows the 64 inputs applied to ADC at each sampling cycle. Each 64 inputs corresponds to CS multiplexor output (Product of input vector with random number). Fig. 4.9 shows the ADC and accumulator outputs at each conversion cycles. For a particular

case study, as shown in the figure, an ideal averaging without quantization results in a output of 270.11mV. The proposed ADC after accumulated 4 samples each provides an output of 269.53mV which is less than 1LSB of error. Fig. 4.10 shows the SNDR of the proposed ADC from input frequency range of 0.2MHz to to 0.98MHz. The ENOB at Nyquist frequency is 7.56. This ENOB achieves Walden FOM [25] of 85fJ/conv. step. Fig. 4.11 shows the DNL and INL of the proposed ADC across 256 digital codes. The worst case DNL is within 0.4LSB. INL is within 1LSB across all digital codes.

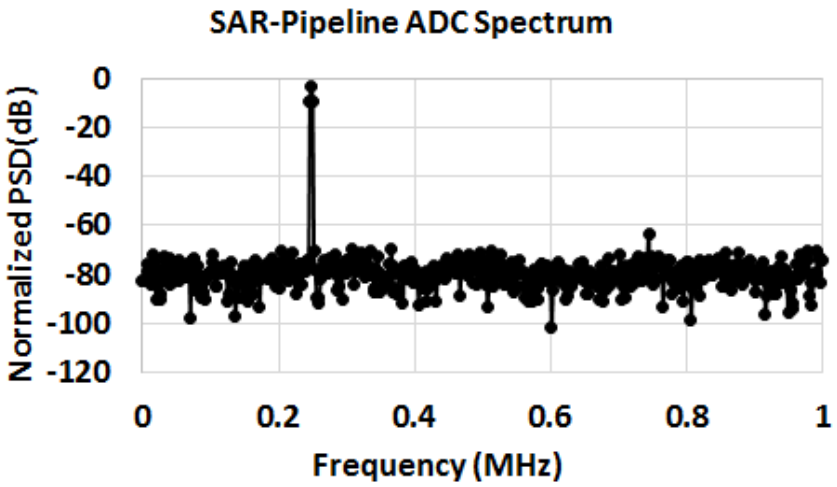


Figure 4.7: Frequency spectrum of the proposed ADC (Fin=248.34kHz & Fs=2MHz)

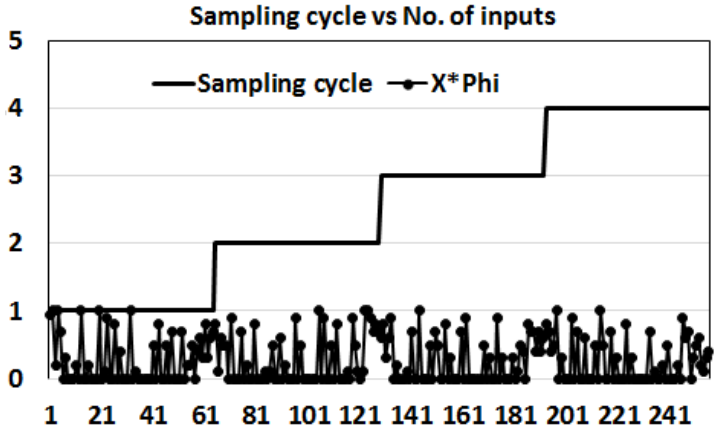


Figure 4.8: 64 inputs for ADC for every 1 sampling cycle

The power budget for the proposed ADC is given in Table .4.2. Even though the total

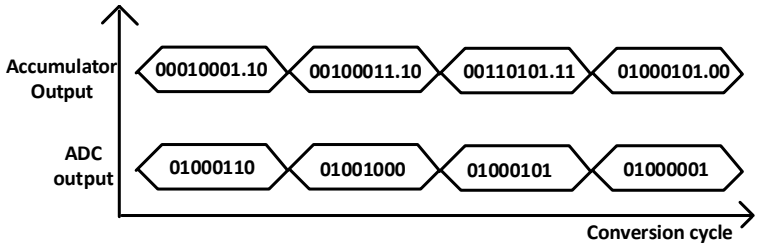


Figure 4.9: Output of ADC and accumulator for 4 conversion cycle

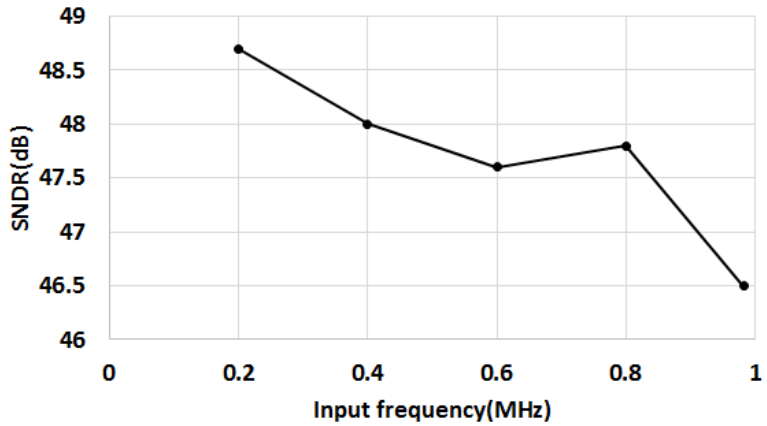


Figure 4.10: Simulation result of SNDR vs Input frequency at Fs=2MHz

power consumed from the supply is 50μW, since the amplifier is shared between two ADC, the power for individual ADC's is 31μW.

Table 4.2: Power and capacitance contribution from individual blocks

Block	Power/	Capacitance
SAR Stage1	7μW	28C
Amplifier	41μW	4C
SAR Stage1	2.5μW	10C
Accumulator	0.5μW	Nil

Table. 4.3 shows the comparison of the proposed design with state of the art CS architecture. Proposed design is scalable and can handle a large number of inputs at the same time.

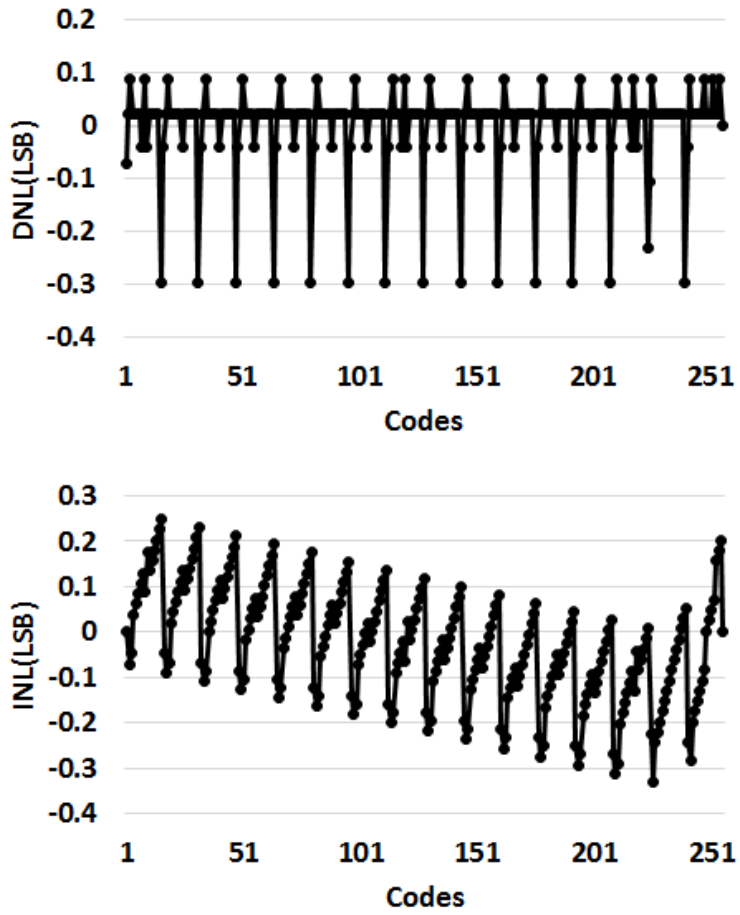


Figure 4.11: DNL and INL of the proposed ADC across 256 codes

Table 4.3: Comparison with reported works

	Oike [21]	Guo [40]	Chen [20]	This work
ADC type	$\Sigma - \Delta$	SAR	SAR	SAR-Pipeline
Technology	0.15μm	0.13μm	0.09μm	0.13μm
Design	Measured	Simulated	Measured	Simulated
No. of inputs	1	4	1	64
Sampling cycles	256	1	256	4
F_s	NA	1MHz	2kHz	2MHz
Capacitance	NA	272C	256C	40C
Power	NA	50μW	5μW	31μW

4.6 Compressed sensing using time based ADC

Fig. 4.12 (a) shows the proposed differential circuit topology for time-based compressive sensing. The input voltage is fed to a ring VCO. The VCO produces a frequency propor-

tional to the differential input voltage ($V_{in} = V_p - V_n$) and is given by:

$$F_t = K_{VCO} * V_{in}(t) \tag{4.7}$$

where K_{VCO} is the linearity coefficient of the VCO. The VCO output drives an up-down 10bit counter. The counter value is proportional to the input voltage. A PRBS sequence acts as the sign for the counter. 0 of PRBS sequence indicates down count and 1 indicates an up counting. The counter in itself acts as an accumulator. The accumulated digital value of the counter for i^{th} measurement (for CR=3) is given:

$$y_i = \sum_{k=1}^{3300} K_{VCO} * \Phi_i(+1, -1) * V_{in}(t_k) \tag{4.8}$$

where $k = 3300$ is the number of conversion cycles. $V_{in}(t_k)$ is the input voltage at k^{th} interval of time. The VCO phase is fed to M number of counters for parallel compressive measurements.

Parallel Compressive Measurement Fig. 5.4 illustrates the details of the time based PRBS and input mixing circuit. We convert input voltage to current using a V to I converter. The V to I converter acts as a current source to the VCO. Hence, the frequency of the VCO is directly proportional to the input voltage in the linear range of the VCO $(250mV - 500mV)$. With M number of PRBS generators and counters we have M different compressive measurements in a given time interval. The matrix multiplication is a true mixed signal (MS) time domain implementation where digital PRBS signals are combined with the absolute value of analog sensor inputs. The proposed time based CS ADC does compressive sensing in (3300 conversion cycle) with M parallel processing units.

Input range extension:

The range of input voltages over which a VCO is linear is limited. Fig. 4.14 shows the proposed input range extension circuit. Fig. 4.15 (a) shows the measured single ended VCO transfer characteristic. The single ended VCO has a linear range of 250mV. Fig. 4.15 (b)

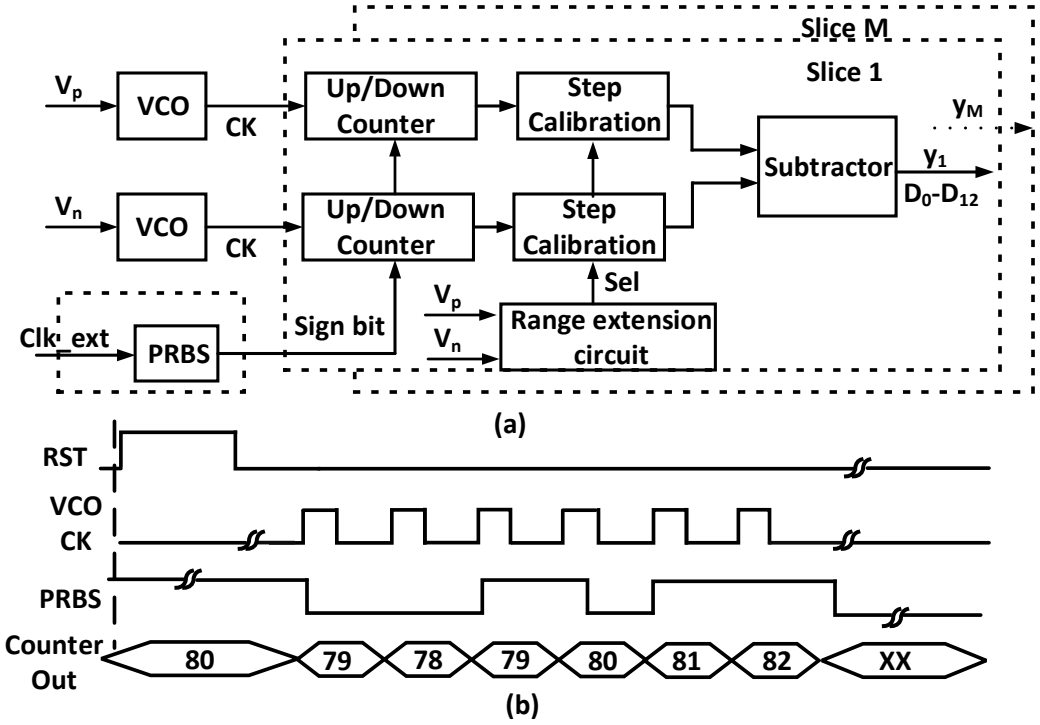


Figure 4.12: (a) Proposed differential architecture for compressive sensing using time based technique (b) Timing diagram for the proposed circuit

shows the differential VCO transfer characteristic. The differential VCO has a linear range of 400mV. After 700mV on either side the slope of VCO transfer characteristic drastically reduces. Therefore, we propose to scale the output by a factor of 4 based on the input level. The proposed circuit uses a clocked comparator which outputs high if the input voltage goes above 700mV, otherwise it stays at 0. This allows us to avoid the saturation of the VCO and extend the operating range by simply scaling the digital output step. The die photo and chip characteristics are shown in Fig. 5.6.

4.7 Experimental Results

Fig. 5.22 (a) shows the DNL of the proposed CS time-domain ADC. The DNL is less than 1.5LSB. Fig. 5.22 (b) shows INL of the proposed CS time-domain ADC. The INL is less than 7LSB. High INL is caused for non-linearities of V-I converter and VCO. Even with an INL of 7LSB, we achieve high classification accuracy via a non-linearity aware

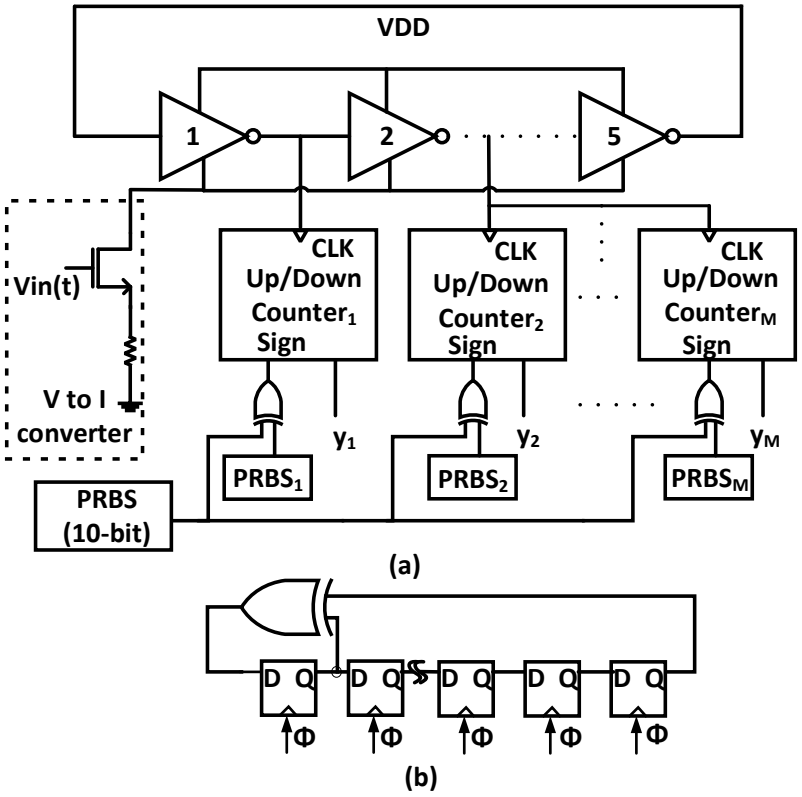


Figure 4.13: (a) Time based PRBS and input value mixing circuit (b) PRBS generator

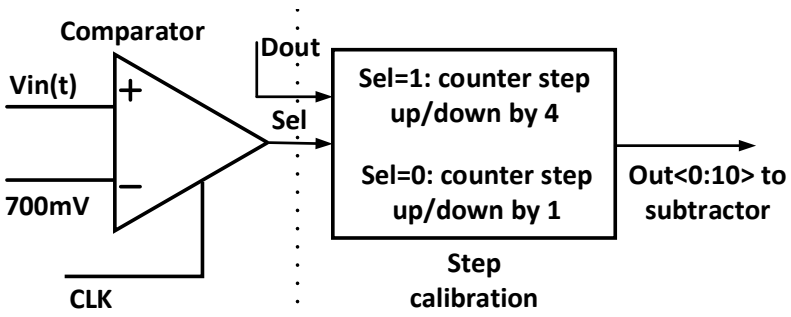


Figure 4.14: Input range extension and calibration via scaling of the counter output based on the input level

training and classification, as explained below. Table 5.2 shows the power drawn by the various blocks of the proposed ADC at an operating voltage of 0.5V for a single bit slice. The VCO is common for all the slices. For a 250nsec pulse width of PRBS (4MHz of operation), the proposed design has $1.25pJ$ of energy for M=160 parallel processing units. Fig. 4.17 (a) shows the VCO frequency of 400kHz at 360mV. Fig. 4.17 (b) shows the VCO frequency of 1.2MHz at 430mV. Fig. 4.18 (a) shows the digital code vs differential

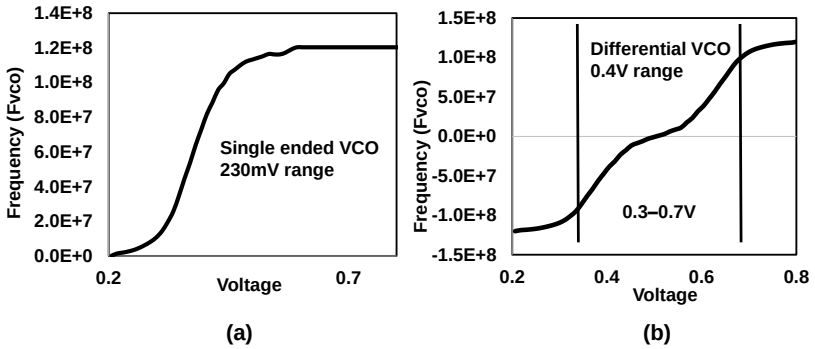


Figure 4.15: (a) Measured single ended VCO transfer characteristic (b) Measured differential VCO transfer characteristic

input voltage of the CS-ADC with and without the calibration for range-extension. Without calibration the slope of the curve after 200mV is 4X less compared to slope before 200mV (400mV range). With calibration the slope of the curve after 200mV is comparable to slope before 200mV (600mV range in total). Fig. 4.18 (b) shows the energy (normalized to the number of parallel multiply and accumulate (MAC) units) and energy (CS for 3300 beats vs no. of parallel processing units). As the number of parallel processing units goes up, the VCO power becomes less significant compared to the counters and the PRBS generators.

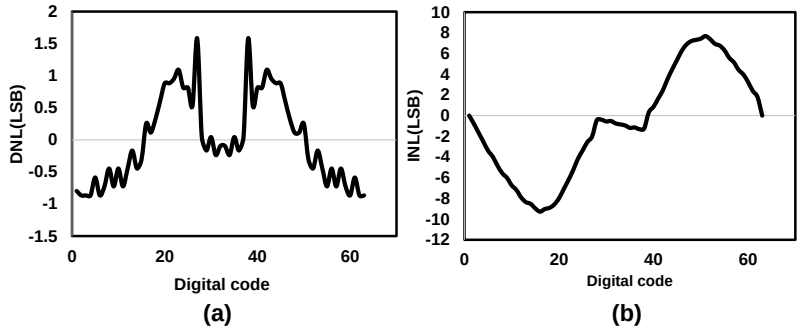


Figure 4.16: (a) Measured DNL of the proposed front-end, (b) Measured INL of the proposed front-end

Fig. 5.23 (a) illustrates the classification accuracy vs compression ratio. Fig. 5.23 (b) illustrates energy (nJ) vs compression ratio. At 8x compression ratio the energy is 10.5nJ for classification (with 2.5% accuracy reduction compared to no compression) . Fig. 4.20 (a) shows proposed technique which includes INL of the ADC while training and testing.

Table 4.4: Measured power drawn by various components

Component	Power
VCO	500nA
Counter	20nA
PRBS	7nA
Adder/Sub-tractor	4nA

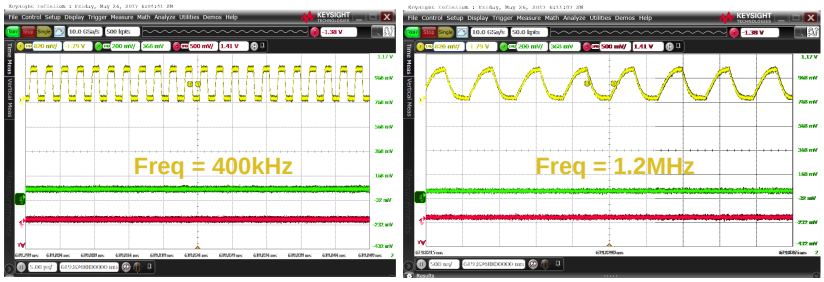


Figure 4.17: Scope captures: a) VCO frequency at 360mV b) VCO frequency at 430mV

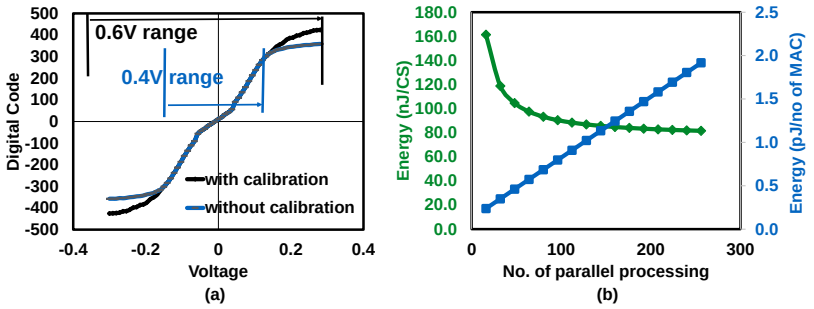


Figure 4.18: a) Measured digital code vs differential input (V_{in}) without and with calibration b) Energy (MAC, CS) vs No. of parallel processing units

Fig. 4.20 (b) shows classification error vs INL with and without INL-aware training. We can observe that INL aware training maintains error level within 2%. We note that the time-domain computational ADC even with significantly high non-linearity can achieve high accuracy if the training is performed considering the INL. We assume front-end amplifier with gain of 100 to be present before ADC. The proposed design shows competitive figures of- merit in terms of power, lower supply voltage when compared to previously published works on CS encoder (Table. 4.5).

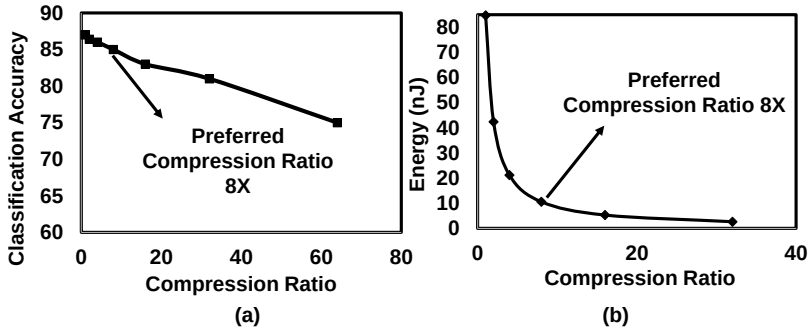


Figure 4.19: a) Classification accuracy vs Compression Ratio b) Energy vs Compression Ratio

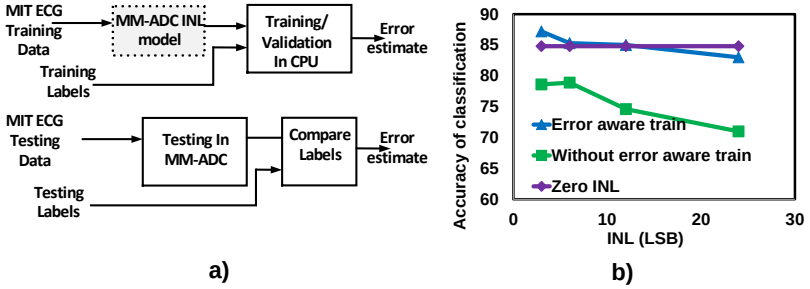


Figure 4.20: a) INL aware training and classification b) Classification accuracy vs INL

Table 4.5: Comparison with state of the art works on CS-ADCs

	Oike	Guo	TCAS-I	TBCAS	Proposed work
System	Image sensing	Radar	EEG classification	EEG classification	ECG classification
Technology	180nm	130nm	130nm	180nm	65nm
A/D rate ENOB	1MS/s 12	1MS/s 10	NA	10kS/s	500kS/s 6bits
Supply	1.2V	1V	1V	1/0.6V	0.5V
No. of parallel processing	1	4	1	Nil	160
Capacitance	NA	272C	NA	516C	0
Power (mu/MHz)	66.3	58	260	13	1.3
Energy (pJ/MAC)	16	0.12	2	3.2	1.47
Area /CS channel	15μm*15μm	NA	10μm*15μm	4μm*200μm	5μm*50μm

CHAPTER 5

VOLTAGE BASED MATRIX MULTIPLYING ADCS

5.1 Smashed filter based classification

The proposed design leverages recent advances in compressive data processing, where it has been shown that a manifold can retain its inherent structure in the CD with very high probability [31]. In particular, given a set of high dimensional vectors

$$\{D_i, X_1, X_2, X_3 \in R^N\}$$

lying on a manifold, the order of the distances between different vectors can be maintained after random linear projections into a space of much lower dimensionality (Fig. 5.1 (a)). This structure preserving property allows us to directly apply nearest-neighbor classifiers in the CD, reducing the overall computation (and hence, power consumption) without loss of classification accuracy. We perform motion center estimation/ moving-object localization in the CD, by judiciously embedding computation in the mixed-signal and digital domains.

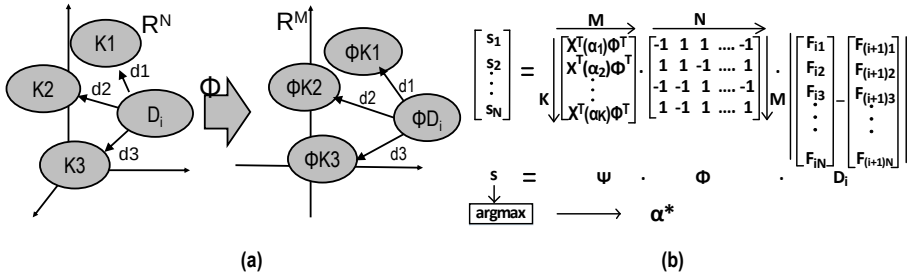


Figure 5.1: (a) Inherent structure of a manifold is preserved in the CD (b) Mathematical representation of the smash-filter operation

Difference images are capable of capturing the motions. Denote F_i as the ith vectorized full resolution image output from the camera containing N pixels in total. Vectorized

difference image D_i of two consecutive frames is calculated as $D_i = |F_{i+1} - F_i|$. To transfer D_i into the CD, we construct a random matrix Φ of size $M \times N$. Each entry of Φ is uniformly chosen from $\{+1, -1\}$ at random. The projection of the vectorized difference image in the compressed domain is calculated as:

$$\hat{D}_i = \Phi D_i \tag{5.1}$$

$\hat{\alpha}^* \approx \alpha^*$ with high probability for some $M \ll N$. By normalizing all the templates to have the same energy, we can further write motion center estimation as solving:

$$\hat{\alpha}^* = \arg\max_{\alpha} X^T(\alpha)\Phi^T\hat{D}_i = \arg\max_{\alpha} \Psi_{\alpha:}\Phi D_i \tag{5.2}$$

The above process is known as smashed filtering [22] which is akin to matched filtering in the CD. It can be decomposed into multiplication with a compression matrix Φ, a smashed filter matrix Ψ, followed by WTA (Fig. 5.1 (b)).

Design Partitioning Between Digital and MS: The proposed algorithm includes multiplication of the input with random number sequences and followed by smashed filter coefficients, followed by accumulation (Fig. 5.1). We analyze the tradeoff between the classification accuracy vs Effective Number of Bits (ENOB) used to represent the multiplier and accumulator outputs.

Fig. 5.2 (a) illustrates that for compressive sensing using random binary matrix both multiplier and accumulator can be within 5-6 bits. Fig. 5.2 (b) illustrates that 80% (90%) accuracy is achieved with 5b (7b) multiplier ENOB but the accumulator needs to be at least 10b wide. We compare the power consumed in a MS vs. an all-digital implementation [44] and note: (1) for ENOB of 5-7, MS multipliers allow $> 6X$ reduction in power (lower process C_{MIN} lowers power consumption but increases linearity requirements at higher ENOB) and (2) high dynamic range requirement in accumulators make digital implementations $> 12X$ energy efficient at ENOB of 10. This allows us to make judicious design

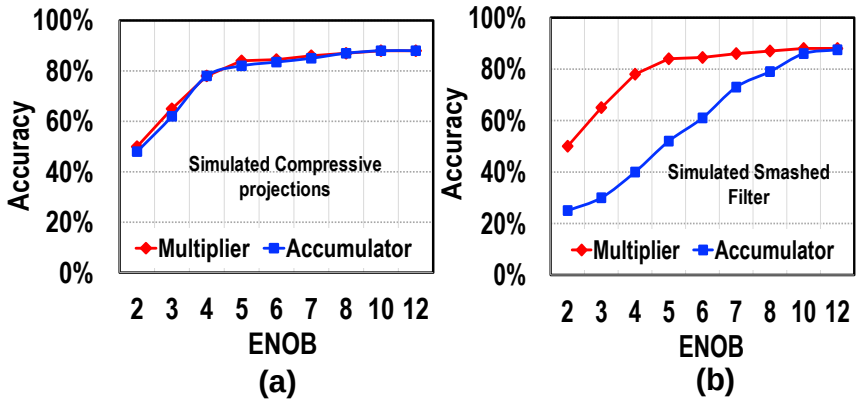


Figure 5.2: Accuracy vs ENOB trade off for (a) compressive sensing and (b) smashed filters

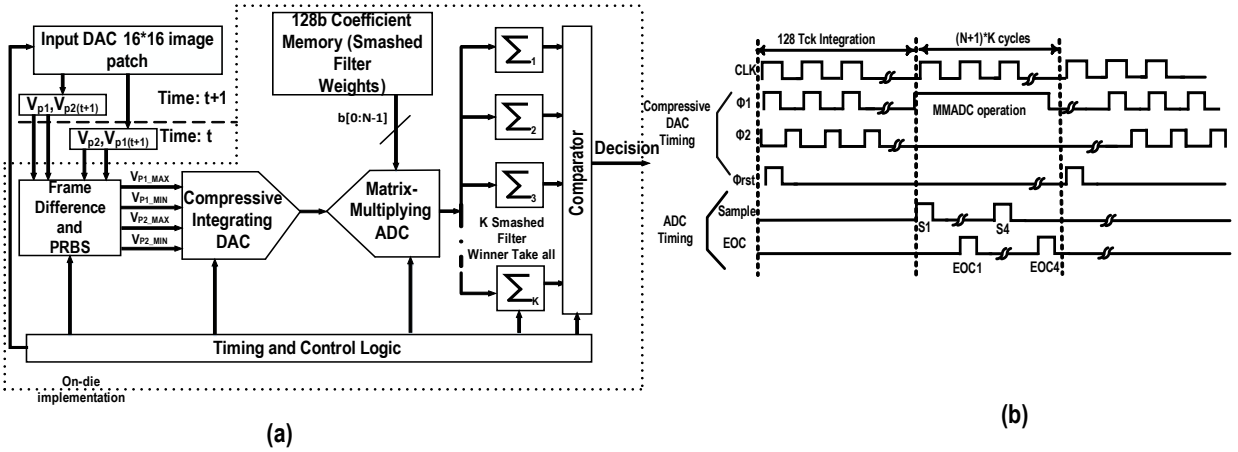


Figure 5.3: (a) Proposed system architecture (b) Timing diagram for the proposed system

partition where the MMADCs enable MS multiplication followed by a digital accumulation and WTA. It should be noted that passive charge sharing multipliers having capacitors lower than 5fF (ENOB 4-5 bits) have low energy/operation compared to traditional digital multipliers [44]. However charge sharing accumulators require amplifier with high gain, higher supply to enhance the dynamic range to more than 5-6 bits, which makes them in feasible for the proposed application (required ENOB is 10).

We propose compressed domain mixed mode classification system. The proposed re configurable system is shown in Fig. 5.3.

5.2 Hardware Implementation

Fig. 5.3 illustrates the top level system architecture and the timing diagram of control signals as implemented in a 130nm test-chip. Frame-data is serialized and applied through an FPGA interface. The current implementation allows four parallel inputs (two from frame t and two from frame t + 1). This represents a slice which can be parallelized in a video application processor. Key design blocks are described below.

Multi-input Compressive Sampling and Integrating DAC: We propose a novel multi-input integrating DAC for in-situ compressive sensing. In the first stage: (1) a comparator identifies max($V_{P(t)}, V_{P(t+1)}$, (2) a PRBS generator combines a random sequence from +1, -1 with the comparator output to create V_{P1MAX} and V_{P1_MIN} (Fig. 5.4). The truth table encoding the PRBS, absolute input difference is shown in Fig. 5.4. Mixed signal implementation reduces encoding power compared to a purely digital encoding and it requires less hardware resources for comparison and finding the absolute value. The output of PRBS and absolute value circuit is given by:

$$V_{P1MAX} = Phi * max(V_{P(t)}, V_{P(t+1)}) \tag{5.3}$$

and

$$V_{P1MIN} = Phi * min(V_{P(t)}, V_{P(t+1)}) \tag{5.4}$$

This acts as input to an integrating DAC (Fig. 5.4 (b)). We propose multi-input integrating DAC as opposed to single input integrating DAC reported in [45]. Generally, camera front-ends require a programmable gain amplifier. We use the programmable gain amplifier as an integrating DAC. The test chip consists of 4 input integrating DAC. Multi-input integrating DAC helps to increase the throughput of the system by a factor of K. where K is the number of inputs to integrating DAC. Capacitance C is the gain setting capacitor used to control the dynamic range at the output of the integrator. A re configurable OTA draws 14A (80A) of current for 5b (7b) operation. For an image patch of $16 * 16$, compressive

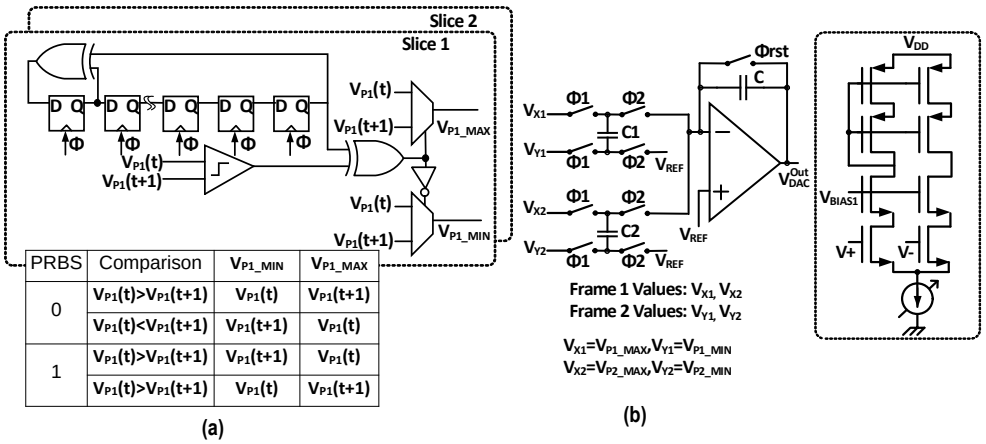


Figure 5.4: a) Mixed signal PRBS and absolute value mixing circuit b) Compressive integrating DAC and reconfigurable OTA

sampling and integration is performed over 128 cycles and the output is given by:

$$V_{out}^{DAC} = \sum_{i=1}^{128} \frac{C_i}{C} * \Phi * abs(V_{Pi}(t) - V_{Pi}(t+1)) \tag{5.5}$$

The matrix multiplication is a true MS implementation where digital PRBS signals are combined with the absolute difference value of analog sensor inputs.

Fixed point Matrix Multiplying ADC: Smashed filter operation is performed in a MM-ADC which operates on V_{OUT}^{DAC}. Smashed filter co-efficient (b_i, 5b or 7b) select V_{OUT}^{DAC} or 0 and the sampled voltage across the capacitive DAC of the SAR ADC is given by:

$$Vout_{samp} = \frac{1}{32} \sum_{i=0}^{4or6} (2^i b_i V_{OUT}^{DAC}) \tag{5.6}$$

Initial investigations reveal that the application requires 5-7b ENOB and justifies the use of a fixed-point SAR ADC architecture. Smashed filter weights are stored in a 128b on-die memory array. The MMADC output is accumulated over K accumulators and a digital WTA identifies match/no-match with template patterns. The entire design supports full-scan and internal nodes are exposed to high-speed pads for test and debug. Die photo and chip characteristics are shown in Fig. 5.6.

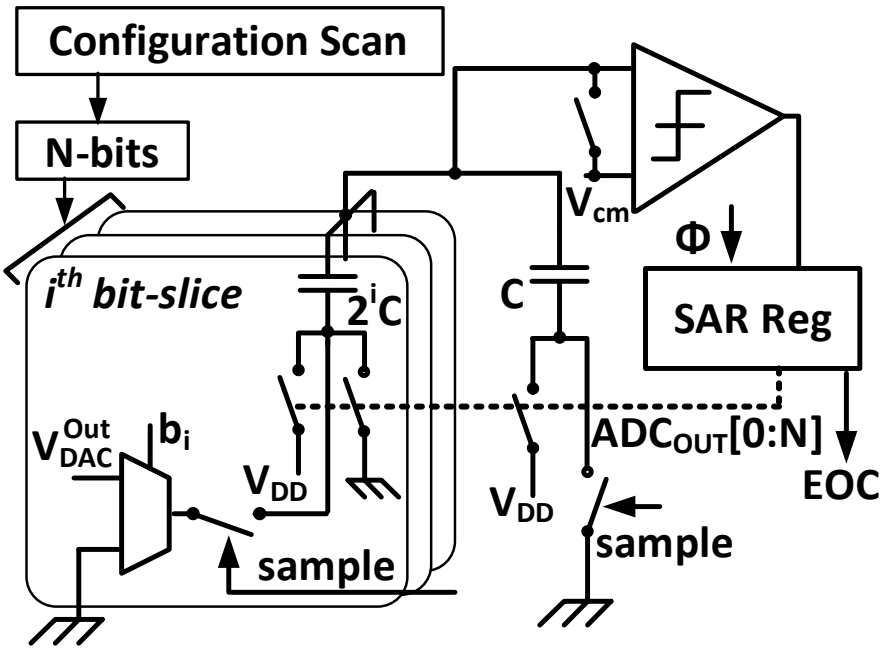
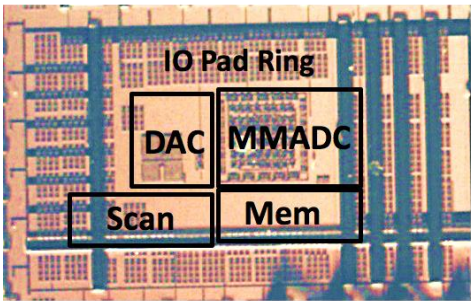


Figure 5.5: Matrix multiplying ADC



Process	GF130nm CMOS 8M
Total Area	2mm²
Active Area	0.6mm²
MMADC	0.113mm²
I/DAC	0.08mm²
Memory, WTA	0.36mm²
Packaging	LCC
No of pads	32

Figure 5.6: Die picture of the chip

5.3 Experimental Results

Fig. 5.7 shows the FFT output of the ADC in the high accuracy 7b mode and the ENOB as a function of the signal frequency at a sampling rate of 1MHz. SNDR of 39.6dB is noted.

Fig. 5.8 shows the oscilloscope capture of the integrator and ADC outputs showing different phases of operation. Here the ADC is configured to operated in the 5 bit mode

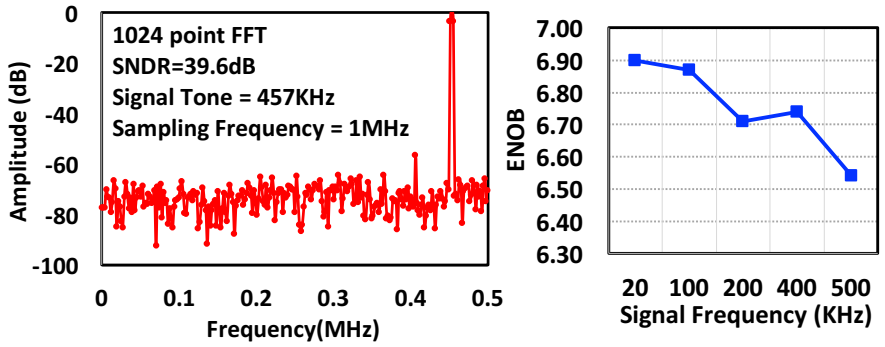


Figure 5.7: Measured ENOB and SNR of the proposed ADC

and number of localization centers is 16. We observe that while integrator does random compression, ADC outputs are zero. When ADC does fixed point matrix multiplication the integrator output stays constant. A typical ADC output of 5b10010 corresponding to 562.5mV is shown (561mV is the integrator output showing error within 0.5LSB).

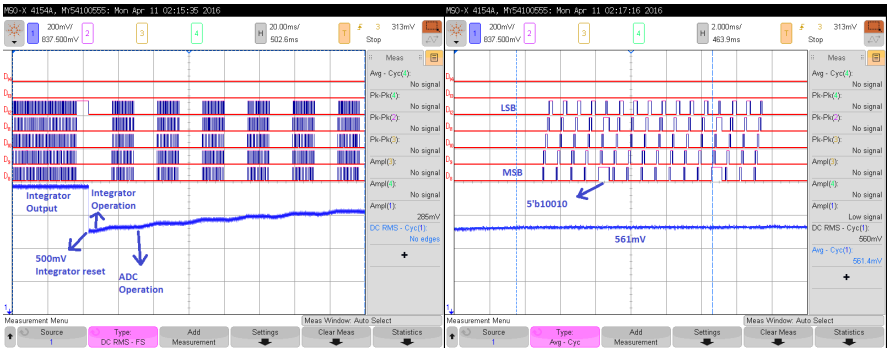


Figure 5.8: Oscilloscope capture of Integrator and ADC outputs

Measured energy-efficiency showed in Fig. 5.9 has 2108nJ/frame (HA mode) and 165nJ/frame (LP) mode illustrating a $248X$ improvement in energy efficiency compared to a baseline all-digital design using matched filters in the pixel domain. We estimate that 16X energy reduction comes from compressive sensing. Another 4X energy reduction is achieved by multi-input compressive DAC. 3.8X energy efficiency is achieved by using 5 bits of resolution in MMADC as opposed to a 12b digital data-path. Fig. 5.23 illustrates the true positive and true negative rates greater than 80% (90%) in 5b (7b) mode. The proposed design shows a competitive figure of- merit when compared to previously published results (Table. I).

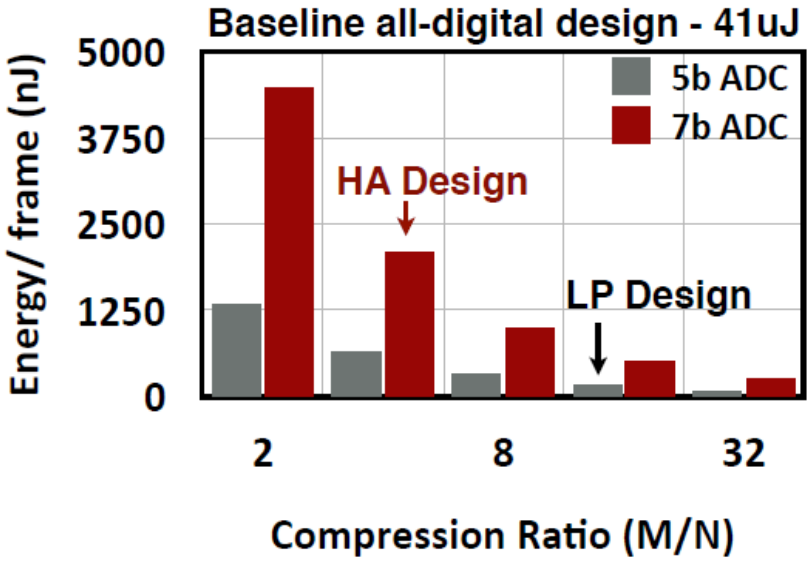


Figure 5.9: Measured Energy of the proposed System

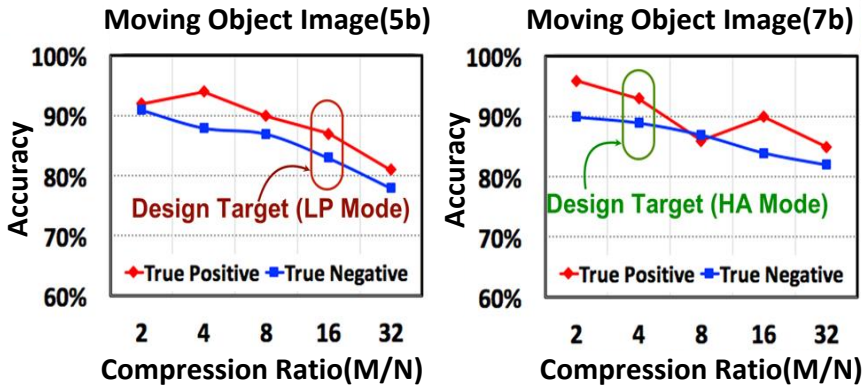


Figure 5.10: Recognition accuracy for object classification and object location in a frame difference image

5.4 Time based matrix multiplying ADC's

Time based ADC's are mostly digital. Porting from one technology to another is easier and they are friendly to both voltage and technology scaling. We propose a time based mixed signal matrix multiplying ADC where : (1) the Digital to time converter produces a pulse proportional to the digital code and (2) the pulse is used to gate the VCO which produces a frequency proportional to the input voltage (Vin) (3) the counter produces a

Table 5.1: Comparison with state of the art works on voltage mode MM-ADCs

	ISSCC 2015	ISSCC 2016	TCAS-I	TCAS-II	Proposed work
System	Object/ECG detection	Object detection	EEG classification	CS ADC	Object/position detection
Technology	130nm	40nm	65nm	28nm	130nm
Detection Algorithm	Ada-boost	CNN	Compressed SVM	CS Recovery	Smashed filter
Energy/ classification	1.38μJ	0.08μJ	3.28mJ (cr=1) 0.13mJ (cr=16)	-	1.3μJ (cr=1) 0.16μJ (cr=16)
A/D rate ENOB	20kS/s 7bits	39MS/s 5-7bits	- -	4GS/s 6 bits	1MS/s 5-7bits
Resolution	Analog/4/8b	Analog 3/6b	-	Analog 6b	Analog 5/7b
TPR/ TNR	0.83/0.89	0.86(overall)	0.96/0.91	-	0.87/0.83

digital code proportional to the frequency (4) the digital code is compared to a bias value using a compactor to obtain the final decision. We have also analyzed the effect of INL on the classification accuracy of handwritten images from the MNIST database.

5.5 Adaboost with SVM as the weak classifier

SVM is a linear classifier that performs classification based on the hyper-plane or a set of hyper-planes. Support vectors represent data points close to the hyper-plane [46]. The MNIST database has 60,000 training data and 10,000 testing data. The samples are represented by $(X_{i1}, ...X_{iK})$ where K=60,000. Each image can be represented by a sequence of samples as: $(x_{i1}, ...x_{iN})$ where N=784. For M number of weak classifiers, the labels are represented by: $y_1, ..., y_M$. For SVM and the Adaboost classifier, we use:

$$y_t = sgn(w * X_i - b) \tag{5.7}$$

$$y_f = max(\alpha_1 * y_1 + \alpha_2 * y_2 + ... + \alpha_M * y_M) \tag{5.8}$$

Here the sgn operation indicates the sign of the operand, w is the array of support

vectors, b is the bias value, $\alpha_i = 1/M$ (where, M=28) and y_f is the final classified label. The hardware implementation of the proposed technique is shown in the Fig. 2.5 (b). The sensor input is directly fed to the time-based MM-ADC. It is a true mixed-signal matrix multiplier, where the sensor input is an analog signal and the weights are digital values.

5.6 Hardware Implementation

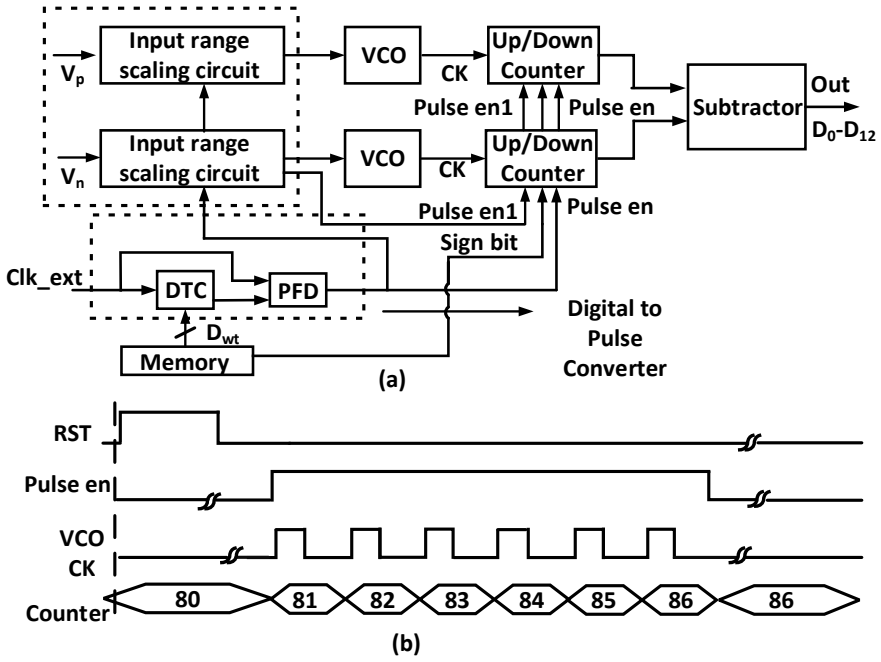


Figure 5.11: (a) Proposed pseudo-differential matrix multiplying time-based ADC (b) The corresponding timing diagram

Fig. 5.11(a) shows the full schematic of the MM-TADC. The frequency of the VCO is directly proportional to the input voltage in the linear range of the VCO. The frequency of the VCO is:

$$F_t = K_{VCO} * V_{in} \tag{5.9}$$

where, K_{VCO} is the linearity co-efficient of the VCO and $V_{in} = V_p - V_n$ is the differential voltage. From the digital weight stored in the on-chip memory, we convert each digital

value into a pulse using a digital-to-time converter (DTC) and a phase-frequency detector (PFD). The DTC receives an external clock reference. The DTC output and the reference clock are fed to the PFD. The PFD produces pulses whose widths are proportional to the digital codes (D_{wt}). The DTC along with the PFD forms a digital-to-pulse converter (DPC). The pulse width of the PFD is:

$$P_t = K_{DTC} * D_{wt} \tag{5.10}$$

The counter gets its clock from the VCO and the enable signal from the PFD output. Since the VCO input is gated with the DPC output, the counter value represents the gated VCO clock. The counter output is:

$$Out1 = K_{VCO} * Vin * K_{DTC} * D_{wt} \tag{5.11}$$

Eq. 5.11 shows that the counter output is proportional to the input voltage and the weights (D_{wt}). The sign-bit is directly fed to the counter. Since the counter is implemented as an up-down counter, we can naturally add or subtract the input multiplied by the magnitude of the weight, depending on the sign of the weight. The counter inherently is an accumulator. It is initially reset to 80 (hex). This is a mixed signal multiplier in the time-domain where the digital weights are multiplied with the absolute difference value of the analog sensor input.

VCO design and input range extension circuits:

Fig. 5.12 a) shows the proposed VCO and the counter architecture. We convert the input voltage to a current using a V to I converter. The V to I converter acts as a current source to the VCO. Two phases of the VCO are used to the feed two counters. $Counter_1$ gets its enable directly from the PFD output (***Pulse-en***). $Counter_2$ gets its enable from the gated PFD output (***Pulse-en1***). Fig. 5.12 b) shows the proposed input range extension circuit. The VCO has a linear range of 250 to 300mV. We propose a technique to extend

the range of the VCO. When the input is more than 600mV, we first scale down the input by $2\times$ and perform analog-to-digital conversion. This preserves the linear operating range of the VCO. Finally, the output is scaled up by $2\times$ to obtain the correct digital code. The proposed circuit uses a clocked comparator whose output is high if the input voltage is above 600mV; otherwise it stays at 0. The VCO receives the input voltage or a zero value from the multiplexer, depending on the *Pulse en* signal from the DPC. Output scaling is achieved as: (a) If the input is less than 600mV, one counter is used to count the VCO frequency (*Pulse-en* always enabled, *Pulse-en1* disabled). (b) Otherwise, two counters are used to count the VCO frequency (*Pulse-en1* is now enabled).

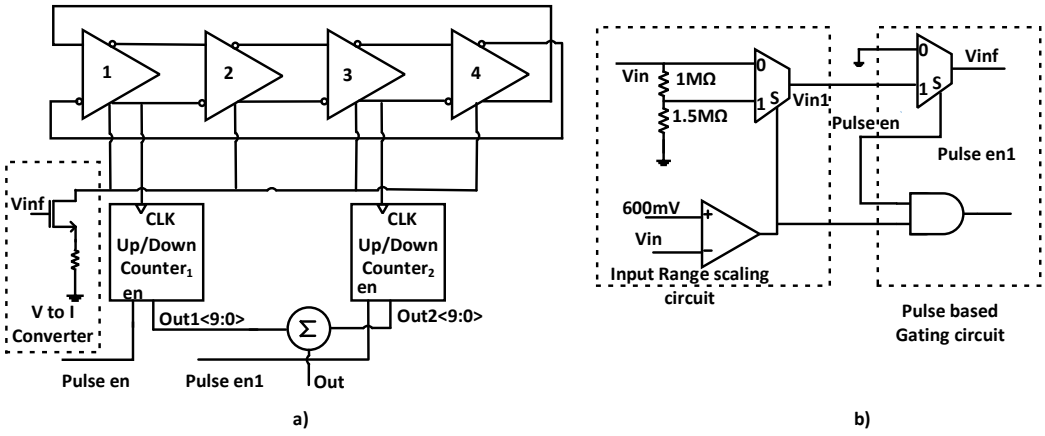


Figure 5.12: a) A single ended VCO along with the output counter b) Input dependent scaling of the output code achieves input range extension

The simulation result of the DTC is shown in Fig. 5.13 with the digital code shown on the X-axis and the corresponding delay shown on the Y-axis. The delay change from minimum code to maximum code is 1μsecs and the DTC exhibits good linearity. The DTC has K_{DPC} of $31n$secs step per digital code.

The simulated linearity plots of proposed DTC is shown in Fig. 5.14 and Fig. 5.15. The differential non-linearity (DNL) and integral non-linearity (INL) are less than 0.5LSB. The proposed DTC exhibits a good linearity performance.The good linearity performance is achieved by common-centroid layout of the inverter chain and programmable capacitors.

Fig. 5.16 shows the digital code vs differential input voltage. Slow Slow (SS) cor-

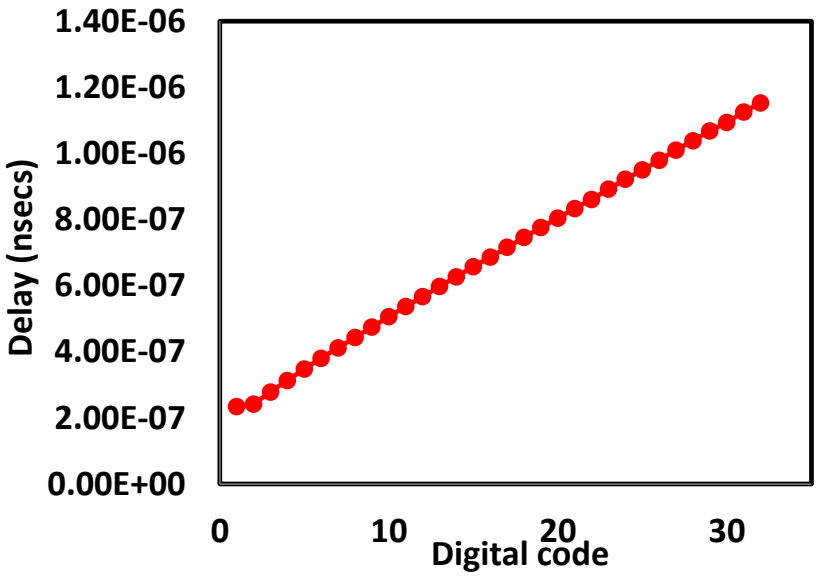


Figure 5.13: Delay of DTC vs Digital code

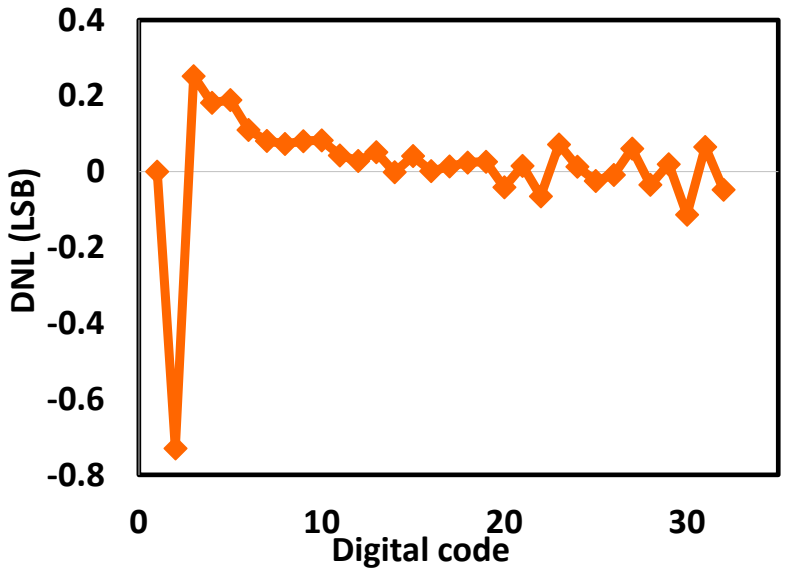


Figure 5.14: DNL of the proposed DTC

ner codes are close to fast (FF) and typical typical (TT) corner. This shows the process tolerance of the proposed time based matrix multiplier.

Fig. 5.17 shows the output of counter for input voltage of VCO (500mV) and mid-code

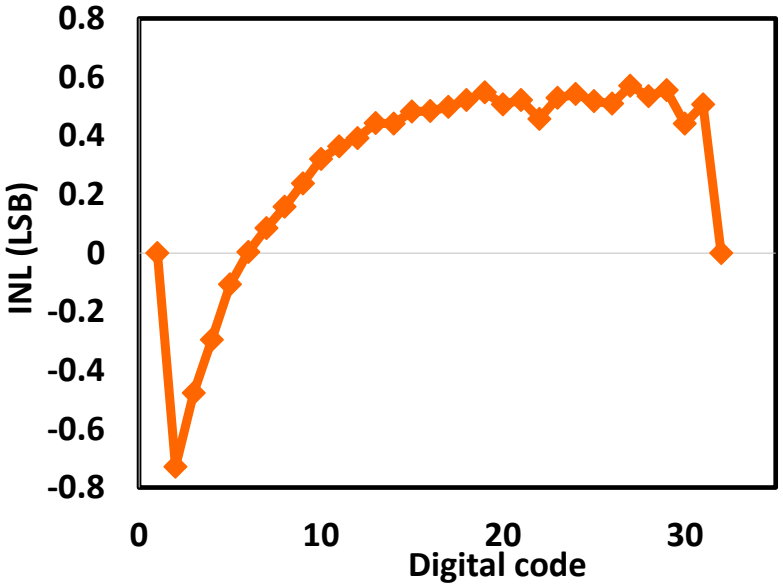


Figure 5.15: INL of the proposed DTC

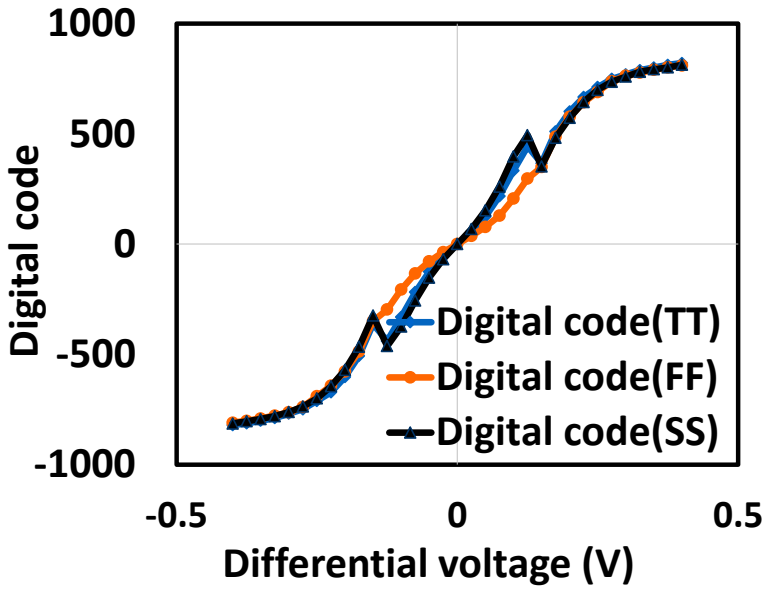


Figure 5.16: Digital code vs Input differential voltage across process variation

of the DPC ($5'b1000$). The peak-peak variation in digital code is 12.

Fig. 5.18 shows the simulated DNL and INL performance across process corners. DNL and INL curves looks similar.

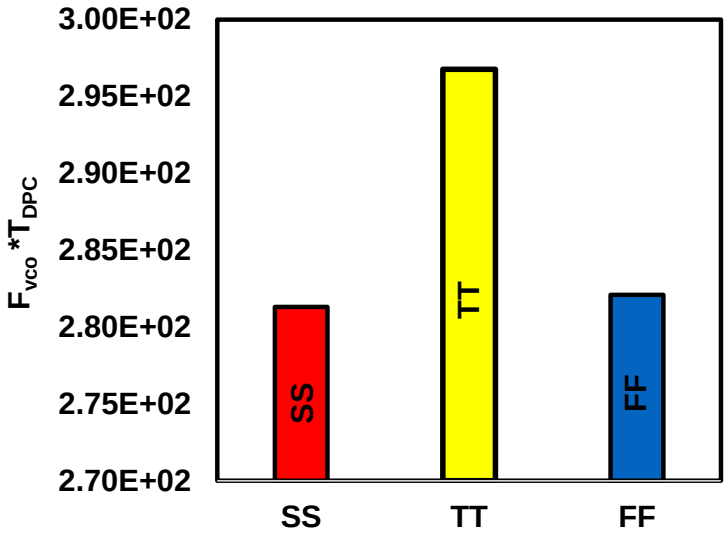


Figure 5.17: Product of VCO frequency and digital code across process corners

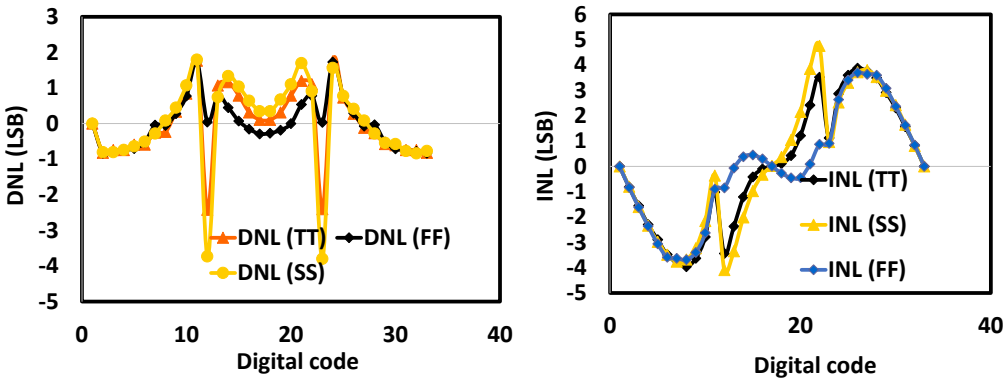
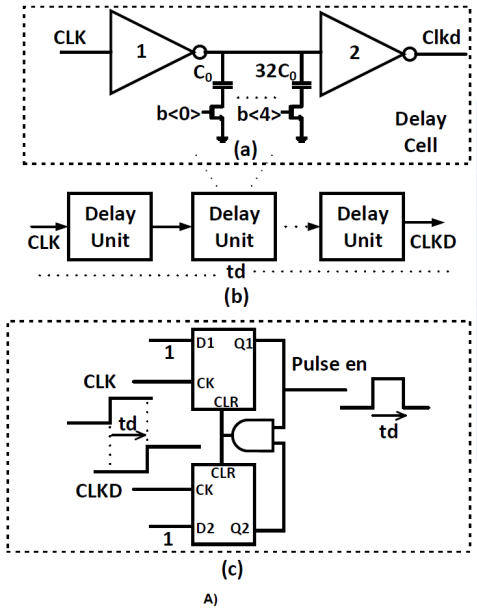
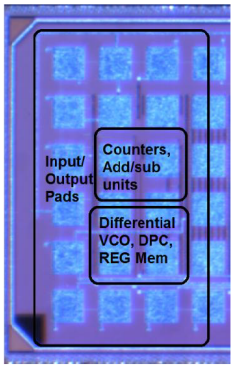


Figure 5.18: Simulated DNL and INL of the proposed ADC across process corner

Digital-to-Pulse converter: The Digital-to-pulse converter (DPC) consists of a DTC followed by a PFD. Fig. 5.19 (b) shows the proposed DTC. It consists of a chain of four delay cells. Each delay cell is a cascade of two inverter cells (Fig. 5.19 (a)), where one inverter has a programmable capacitance as the load. An unit capacitance, C_0 of 10fF is implemented. The DTC receives an external clock. The output of the DTC is represented by CLKD. The delayed clock and the original clock are fed to a PFD. The PFD produces a pulse which is proportional to the delay between CLK and CLKD (Fig. 5.19 (c)). The test-

chip is implemented in 65nm CMOS. We have implemented a bit-slice of the design and input for characterization is serially streamed in for system level analysis. The die photo and the chip characteristics are shown in Fig. 5.6 B).





Process	GF 65nm CMOS 8M
Active Area	0.03mm^2
VCO Area	0.0042mm^2
Packaging	LCC
No. of Pads	32

Figure 5.19: A) Digital to pulse converter (DPC) for pulse generation B) Die shot and chip characteristics

5.7 Experimental Results

Fig. 5.20 (a) shows the traditional training procedure for classification of the MNIST database. The VCO based ADCs have low-power but poor INL performance. Here we show that the training can be aware of such non-ideality. Fig. 5.20 (b) shows the proposed training procedure for the MNIST classification. The proposed technique comprehends the non-linearity (INL) error included in the matrix multiplication. This error-aware model is used for training and testing (Fig. 5.20 (c)) the MNIST images. This error-aware technique helps to improve the accuracy of the classification. Fig. 5.21 (a) shows the measured single-ended VCO transfer characteristics. The single ended VCO has a linear range of 250mV. Fig. 5.21 (b) shows the VCO transfer characteristics with the range extension technique. The proposed extended-range differential-VCO has an input range of 800mV.

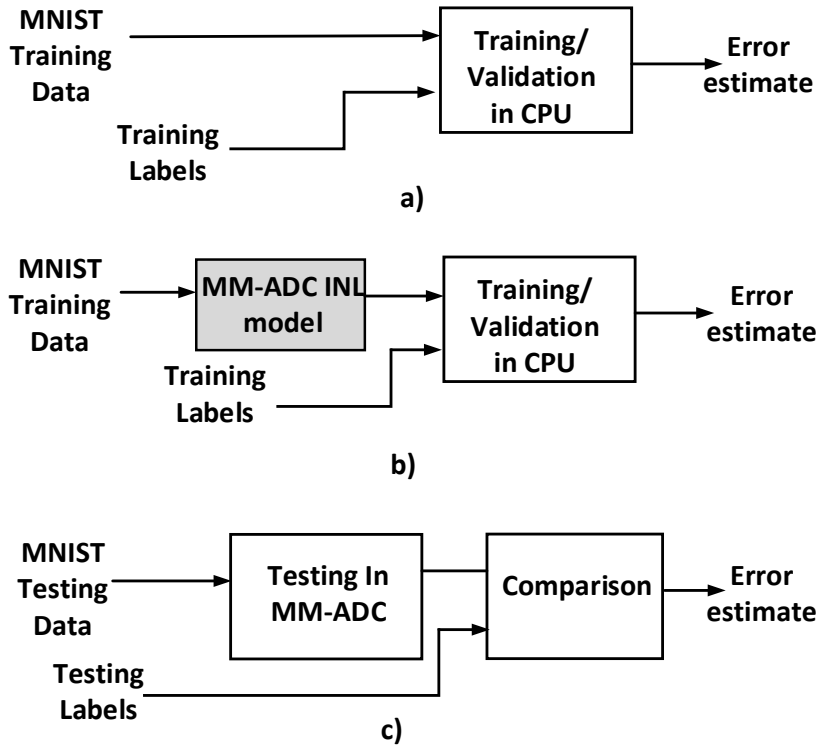


Figure 5.20: (a) Traditional training procedure (b) Proposed INL aware training and (c) Testing for accuracy of inference

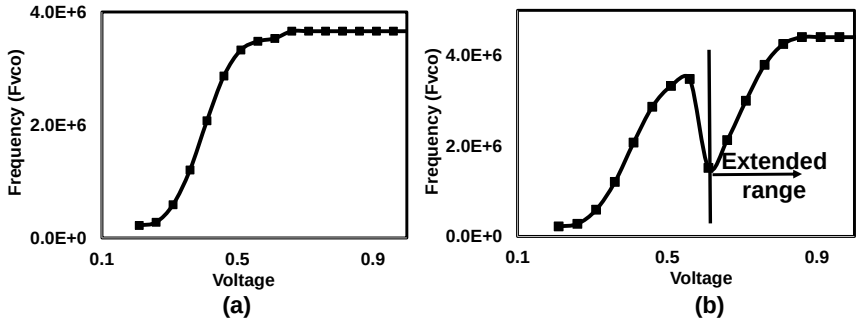


Figure 5.21: Measured: (a) Single ended VCO transfer characteristic (b) Folded VCO transfer characteristic with high input range

Fig. 5.22 (a) shows the DNL of the proposed MM-TADC. The DNL is less than 1.5LSB. Fig. 5.22 (b) shows the INL of the proposed MM-TADC. The INL is less than 7LSB. Fig. 5.23 (b) shows the classification accuracy with and without the INL error-aware training. We can observe that without error-aware training, the classification error goes up with increasing INL. However, with INL error-aware training, the classification

error remains within a $\pm$2-3% margin. The ADC energy per step scales as 2^{2*ENOB}. Here, ENOB of 5.3 can achieve the same accuracy as an 8 bits with a large INL. With the proposed circuit we achieve more than 5X reduction in power using the TD ADC. Fig. 5.23 (a) shows the digital code vs the differential input voltage for the proposed circuit. We can observe that after 600mV, the range gets extended. This is achieved by the proposed folding architecture before the VCO. Table 5.2 shows the power drawn by various blocks of the proposed MM TADC. For a 700ns pulse width of the DPC, energy/MAC is 1.47pJ. For $28 * 28$ image size with 60 support vectors the estimated total energy is 313nJ. For Adaboost with 26 weak classifiers, the energy/classification is 8.1μJ. Table II shows the comparison with previously reported results [13] [47] [14]. [13] [47] perform matrix multiplication in the voltage domain. The amplifier used in [47] requires a 1-1.2V supply. The measured energy of our work is comparable to reported results and it uses a digitally scalable architecture enabled by time-based processing.

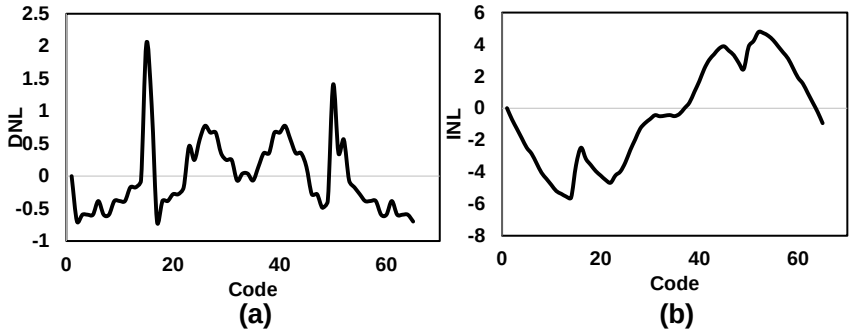


Figure 5.22: (a) Measured DNL of the proposed front-end (b) Measured INL of the proposed front-end

Table 5.2: Power drawn by various components

Component	VCO	Counter	DPC	Adder	Total
Power	360nA	40nA	80nA	16nA	376nA

Fig. 5.24 shows the performance of the time-based ADC with and without INL aware training across process corners. The typical typical (TT) model (SVM co-coefficients) at 25^oC can be used for slow slow (SS) and fast fast (FF) corners. TT-model extracted from

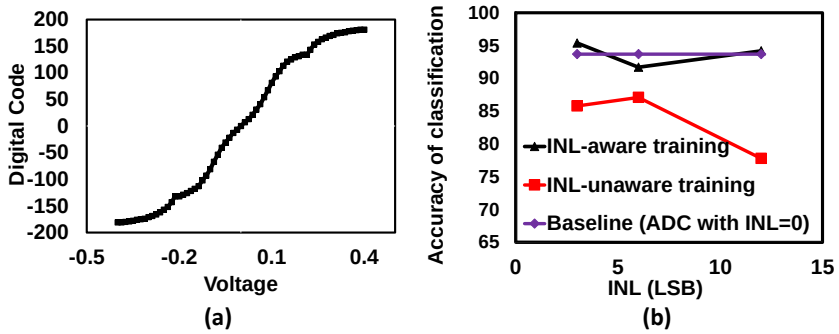


Figure 5.23: (a) Digital code vs differential input voltage (b) Classification accuracy for a baseline design and TD ADC based design with and without error aware training

training can be used for testing SS and FF corner chips as well with accuracy comparable to Zero-INL model.

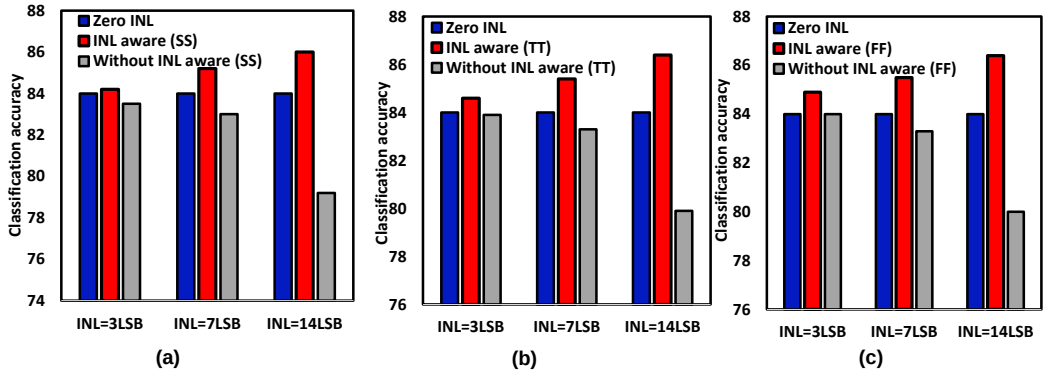


Figure 5.24: Classification accuracy of the proposed time based ADC with and without INL aware training for (a) SS corner (b) TT corner (c) FF corner

Table 5.3: Comparison with state of the art works on MM-ADCs

	ISSCC 2015	ISSCC 2016	ISSCC 2017	ASSCC 2016	Proposed work
System	Object/ECG detection	Object detection	Spatial filtering	Image classification	Image classification
Technology	130nm	40nm	65nm	28nm	65nm
A/D rate ENOB D/A rate	20kS/s 7bits N/A	39MS/s 5-7bits 1GS/s	NA 14bits 1.5MS/s	2.4MS/s 8 bits -	500kS/s 5.4bits -
Multiplication mode	Voltage	Voltage	Voltage	Voltage	Time
Analog/Digital supply	1.2V	1.1V	1V	1V	0.4V
Input range	0.6V	0.5V	1V	1V	0.8V
Power (mu/MHz)	66.3	0.228	260	3	0.46
Energy (pJ/MAC)	16	0.12	2	3.2	1.47
Classification accuracy	87%	-	89%	-	95%

CHAPTER 6

MIXED SIGNAL NEUROMORPHIC ACCELERATOR FOR AUTONOMOUS ROBOTS

Most of the recent hardware demonstrations in deep neural networks (DNNs) and convolutional neural networks (CNNs) have addressed image classification applications [48, 49, 50, 51, 52] and have demonstrated breakthrough improvements in energy-efficiency. However, autonomous systems such as mobile-robots, which continuously interact with the environment need to make decisions. Although this problem shares some similarities with image classification, we understand that truly autonomous systems need to be able to make such decisions in real-time and also learn from its experiences. Hence, the next generation of hardware systems that can enable autonomy in mobile-robots, must enable decision-making such as path-planning, obstacle-avoidance etc. Fig. 6.1 provides a broad overview of the three principal classes of machine learning (ML) paradigms: (1) *Supervised learning:* It involves learning from large sets of labeled data. Typical examples include training image classifiers using the Image-Net database [53] or the MNIST database [54]. Supervised learning involves a training phase and a classification or inference phase. (2) *Unsupervised learning:* In unsupervised learning, the neural network is not trained using labeled data. Instead, the system is trained to create clusters from the training set [55]. This learning approach can be more bio-mimetic and recently it has been used to train spiking neural networks. (3) *Reinforcement learning:* In reinforcement learning, an autonomous agent trains itself in real-time, by interacting with a dynamically changing environment and learning from its many experiences. In a pursuit to learning, the agent first senses the environment using vision sensors, depth sensors, location/position sensors etc. Next it takes an action and calculates a reward based on the action. The reward allows the agent to quantify the so-called quality of its action in the given state-space. Eventually,

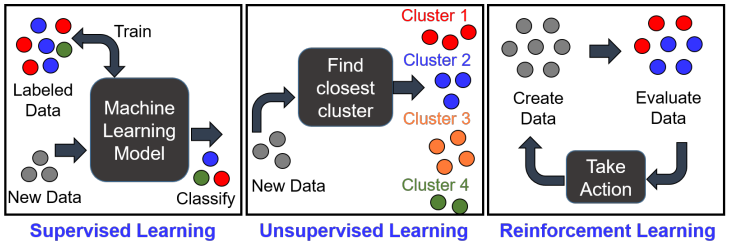


Figure 6.1: Three types of machine learning paradigms (1) Supervised learning (2) Unsupervised learning and (3) Reinforcement learning

it learns to maximize the notion of cumulative reward in the long run. Thus RL is a biologically inspired algorithm that can provide true autonomy in intelligent systems. Recent it has enjoyed considerable success both in control [56] and game-play [57]. Notably, RL and other machine learning techniques were used to beat the best human player in Alpha-Go and play a suite of ATARI games [57].

From a hardware perspective, increased energy-efficiency is a key enabler for true autonomy in edge nodes. As researchers are exploring hardware accelerators for ML applications, it has been shown that in most networks, 5-6 bits of resolution is sufficient to enable accurate and robust inference [13, 28, 14]. This has lead to innovative circuit architectures that target low-power and area [49, 13, 28]. At lower precision, analog and mixed-signal computing blocks have shown promise. In particular, most of the demonstrations have used voltage mode circuits [14, 13], enabled by switched capacitor designs to implement linear algebraic kernels. For example, [27] demonstrates 4-8X improvement in energy-efficiency. However, it is worth noting, that even with the relatively lower precision of 5-6bits, a suitable dynamic range and acceptable linearity need to be maintained. This has resulted in designs where the supply voltage for mixed-signal circuits is limited to 1-1.2V [13, 14]. On the other hand, algorithm-hardware co-design has resulted in efficient digital implementations of dynamic-accuracy-voltage-frequency-scaling (DAVFS) [49], wide and variable precision CNNs [58, 59] and even binary neural networks [58, 60, 61]. However, the applicability of binary networks in successfully solving complex tasks need to be further ascertained. [62, 63] have further demonstrated on-chip learning using neuromorphic

sparse coding techniques. Thus the landscape of energy-efficient ML has seen considerable advances and in particular, voltage-based analog/mixed-signal computing has recently gained well-deserved attention.

To address the problem of supply-scalability in voltage-mode analog circuits, we propose a different approach, namely encoding information and computing in the time-domain. An earlier demonstration of time-domain compressive data-conversion can be found in [64]. Time-domain mixed-signal (TD-MS) designs inherit the advantages of analog-computing including high energy-efficiency at target bit resolutions. However, it also allows us to relax the supply voltage requirement, since the dynamic range is expressed in time. Our proposed time-based multiplication-and-accumulation (MAC) technique provides an energy-efficient alternative to digital implementation, exhibits seamless interfaces with digital memory circuits and also demonstrates voltage and process-scalability like digital logic.

In spite of the success of RL on the algorithm front, hardware demonstrations of RL are limited to early demonstrations in object recognition [65] and noise-shaping [66]. However, for energy-constrained mobile-robots that are used for surveillance and exploration; advances in RL algorithms need to be matched with efficient circuit and hardware designs. In this paper, which is a follow up of [67], we demonstrate a neuromorphic accelerator for RL which: (1) implements Q-learning using a fully connected neural network, (2) demonstrates energy-proportional-computation using TD-MS circuits, and (3) uses a stochastic network of synapses to reduce data over-fitting. The proposed accelerator consumes 690μW of peak power at 1.2V. The accelerator can operate down to a low supply voltage of 0.4V; making it a voltage-scalable mixed-signal neural network.

6.1 Basics of Reinforcement Learning

For a detailed overview of RL techniques, algorithms and applications, interested readers are referred to [56]. In this section, we will provide a short overview of Q-learning which has been implemented in the current design. The problem space in RL consists of agents,

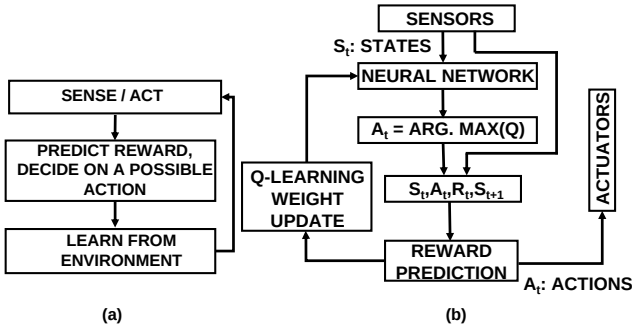


Figure 6.2: (a) The basic steps in RL (b) Q-learning as an implementation of RL that has been implemented in this test-chip

states, actions and cumulative rewards. The agent transitions between states (S_t and S_{t+1}) via an action A_t. From a neurobiology stand-point RL finds similarities when compared to the operation of the basal ganglia in the human brain [68]. It has been observed that the cerebral cortex performs state/action representations, the striatum performs reward predictions, the palladium assists in action selection, and the dopamine neurons perform complex operations akin to the temporal difference method. Q-learning inherits these neuromorphic properties and continues to be a well-studied RL technique. This has been illustrated in Fig. 6.2 (a) and (b), where the different parameters have been shown. Q-learning works on the principle of the action-value-function ($Q(S_t, A_t)$) which can implement an optimal action policy for a finite set of states. This is an off-policy learning technique [56] and has been shown to exhibit faster convergence compared to on-policy learning algorithms such as SARSA [69]. The core of the algorithm is based on the iterative update of the Q-value, as an agent navigates through a series of (state,action,reward) tupules. This iterative scheme is derived from the Bellman equation [70] for optimal control. The iterative algorithm can be summarized as:

$$Q_{t+1}(S_t, A_t) = Q_t(S_t, A_t) + \alpha(R_t + \gamma \max Q_t(S_{t+1}, A_t) - Q_t(S_t, A_t)) \tag{6.1}$$

where, γ is the discount factor and α is the learning rate. α is reduced from 1 to zero in the steps of 0.001. γ is used to prioritize the rewards; immediate rewards are given higher priority whereas the distant rewards are discounted. It is easy to understand that the size

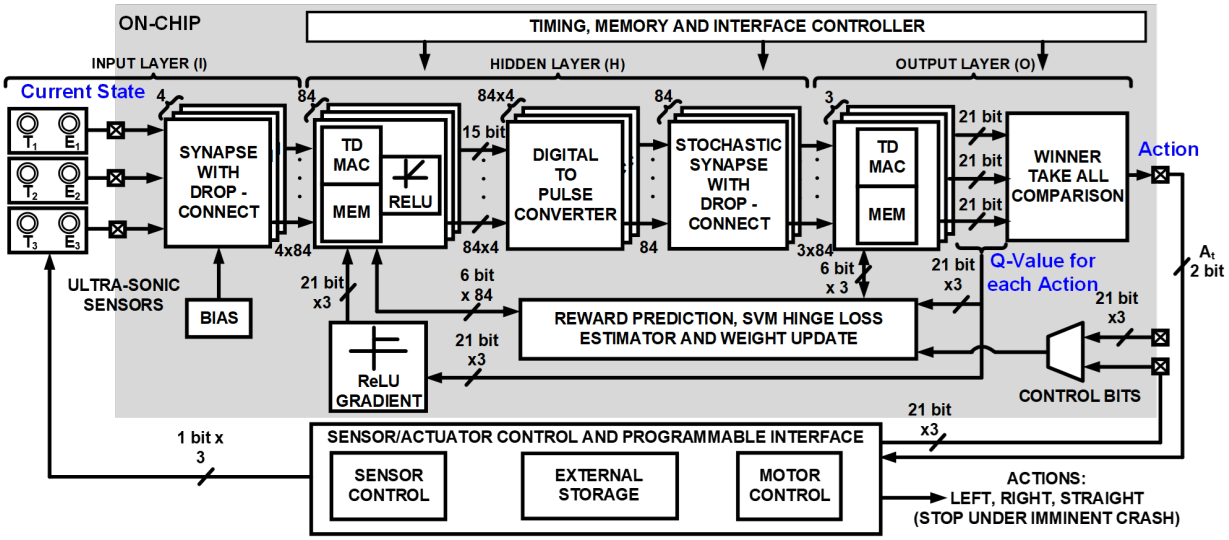


Figure 6.3: System architecture illustrating the different circuit blocks and the interface to the external micro-controllers (Raspberry PI) and motor drivers

of the state-action space grows as $(S \times A)$. To enable real-time Q-learning, one can store the Q-values corresponding to the state, action, reward, and next-state in a look-up table. However, with a growing $(S \times A)$, a huge memory capacity is required. Therefore, the Q-value is typically approximated as a neural network output [57]. The sensor provide the states (S_t). States act as inputs to the neural network. The neural network provides the Q-values corresponding to the set of possible actions. In our demonstration system, we use three ultrasonic sensor inputs, and the mobile-robot can move in three possible directions (left, center or right). Once, the maximum argument of the Q-value is determined, the robot moves in that particular direction with a finite probability, ϵ. The value of ϵ is fixed in the current design. By taking a series of actions in the state-space the robot calculates the reward for each action and trains the neural network via back-propagation, thus creating a robust functional mapping from the state-space to the action-space. The principal steps carried out by the neural network constitute of the following:

1. For the current state S_t, we perform a forward pass of the neural network to obtain the Q-values for all actions.

2. We take an action, A_t perform a forward pass for the next state, S_{t+1} and calculate maximum overall network outputs $max_{A_{t+1}}Q(S_{t+1}, A_{t+1})$.

3. We set the target Q-value as a sum of the immediate reward, R_t and $\gamma max_{A_{t+1}} Q(S_{t+1}, A_{t+1})$, using the maximum value calculated in the previous step.

4. We define a loss function based on the target and update the model weights using backpropagation.

We will describe mapping of the algorithm to the current hardware design in Section 6.4.

6.2 System Components and Architecture

In this section, we provide a brief overview of the system components and the system architecture.

6.2.1 Ultra-sonic Sensors as State Estimates

We use ultra-sonic sensors for measuring the distance of obstacles. The ultra-sonic sensors receive periodic trigger signals from an associated raspberry PI based micro-controller. The sensor emits a sonic signal in response to the trigger signal. The echo pulse is raised to high at the end of the the sonic burst. Once the reflected wave is received, the echo pulse is pulled down. The pulse width of the echo signal is proportional to the distance from the obstacle. The distance from obstacle is evaluated as:

$$d = \frac{t * c}{2} \tag{6.2}$$

where d is the distance from the obstacle, c is the velocity of sound in air, and t is the pulse width of the echo pulse. In traditional digital processing, the echo pulse needs to be converted to a digital value. Hence, it requires a pulse to digital converter. However, in the current TD-MS design, the neural network can directly process the pulse based echo signal thus reducing both the latency and energy of analog (time) to digital conversion. Our current system uses three ultra-sonic sensors and the vector of the three distances

$[d_{left}, d_{center}, d_{right}]$ defines the state-space. The left and right sensors are placed at an angle of 30with respect to the center.

The timing of the ultrasonic sensor is shown in Fig. 6.4 (b).

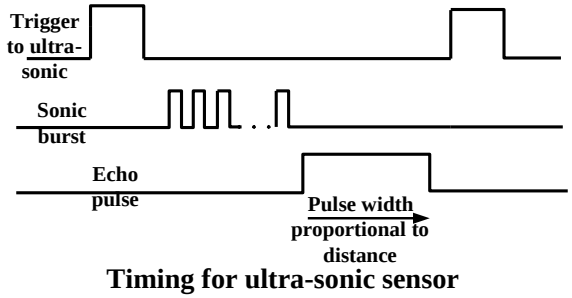


Figure 6.4: (a) Ultra-sonic sensor (b) Timing diagram of ultra-sonic sensor

6.2.2 System Architecture

Fig. 6.3 shows the system architecture and illustrates four main components : (1) three ultra-sonic sensors (2) the proposed RL test-chip (3) a Raspberry PI based micro-controller and (4) motor drivers and motors for each of the four wheels. The ultrasonic sensors are used to find the distances from three directions, as described in the previous section, and is periodically clocked by the Raspberry-PI. The echo signals from the ultrasonic sensors are fed to the test-chip and the Raspberry-PI. During RL training the Raspberry PI stores the (S_t, A_t, R_t, S_{t+1}) in a scratchpad memory and feeds the data to the test-chip. The test-chip first performs inference (from sensed states to actions). Next, it updates the reward-based neural-network model via back-propagation and gradient descent. The test-chip sends a 2-bit control word indicating one of three possible actions (*move left*, *move forward* or *move right*) to the Raspberry-PI, which in-turn sends the control command to the motor controllers.

6.2.3 Neural Network Implementation

A fully-connected (FC), three-layered neural-network provides a functional mapping from the state-space to the Q-values of the action-space. The first layer obtains pulse-domain data from the three ultra-sonic sensors ($D = [d_{left}, d_{center}, d_{right}]$) as well as a bias value for the input layer ($BIAS$). The inner products of (D) and the weights from the input layer to hidden layer (w^{IH}) are computed in the time-domain and accumulated in a 15-bit counter. A digital word, consisting of the first 7 MSBs of the counter, is fed to a digital-to-pulse-converter (DPC). The DPC natively implements a rectified linear unit (ReLU) based activation function and produces neuron outputs as pulses. We choose ReLU, as opposed to other activation functions, because of faster and more stable convergence. The outputs to the DPC represent the outputs of the hidden layer of neurons (H):

$$H = ReLU(D * w^{IH}) \tag{6.3}$$

The number of neurons in the hidden layer is 84. In our demonstration platform, we simulated various grid-world examples and for a grid-world of 10^4 nodes, we found that 84 neurons in the hidden layer performed satisfactorily both in terms of convergence speed and accuracy. More complex environments with more complex dynamics (such as moving obstacles) will require a larger number of neurons and hidden layers. Further, it is worth mentioning that navigation via ultra-sonic sensors has limited accuracy and is also computationally less intensive. At-scale systems that can navigate using camera based images require significantly larger neural networks models. However, the current prototype demonstrates an RL system enabled by scalable hardware primitives, capable of learning to navigate in a real-environment, albeit with limited complexity. The outputs of the hidden layer (H) are multiplied by the synaptic weights of the hidden-to-output layer (w^{HO}) to

produce the final output (Q) of the neural network. We can express this as:

$$Q = H * w^{HO} \tag{6.4}$$

The output (Q) consists of 3 Q-values corresponding to the three directions for the motor control (Q_{left}, Q_{center}, Q_{right}). The argument of the maximum Q-value among the three outputs determines the direction of motor control (A_t) via a winner-take-all (WTA) circuit. Thus the agent can move either left, forward or to the right. The flow of input pulses from the ultra-sonic sensors to the hidden layer outputs and finally to the output of the network is sequential. A_t is encoded in a 2-bit word and sent off-chip to the Raspberry-PI for motor control. We use a scan chain to initialize the neural network weights to random values. Alternatively, the neural-network can be initialized to pre-defined weights obtained from simulations, using transfer learning [56].

6.3 TD-MS Circuit Architecture and Design

In this section we introduce the design of TD-MS circuits and corresponding circuit architectures.

6.3.1 Multiply-and-accumulate (MAC) in a TD-MS architecture

Fig. 6.5(a) illustrates the proposed time-based multiply-and-accumulate (MAC) circuit. It has a pulse input (T_{p_i}) used as the "Enable" signal to an up-down counter. This pulse is obtained from the i^{th} ultrasonic sensor ($i = 1, 2, 3$) in case of the hidden layer neurons or from the i^{th} hidden layer neuron ($i = 1, 2, .., 84$) in the case of the output layer. We store the synaptic weights of all the fan-ins locally at each neuron. The weight is a 6-bit value expressed in signed-magnitude format where the MSB is the sign bit. The 5 LSBs (W[0:4]) are connected to a multiplexor. T_{p_i} controls the select bit of multiplexor. If the digital pulse is high, the 5-bit word is passed through the multiplexor, or else a 0 is passed

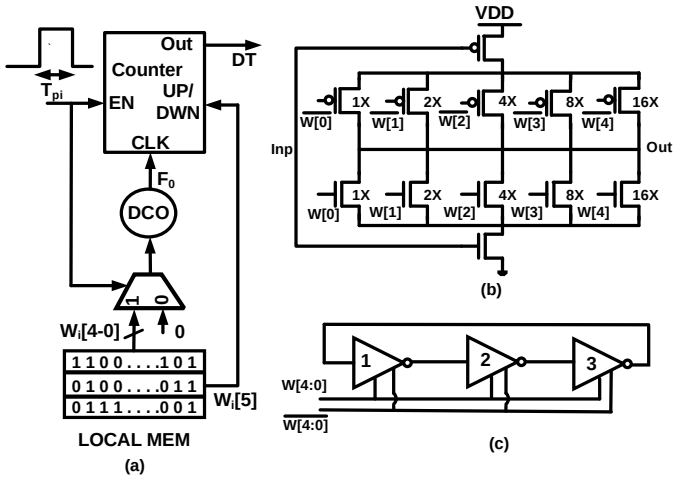


Figure 6.5: (a) Proposed TD-MS multiply and accumulate (MAC) unit, (b) DCO unit cell, (c) 3 stage DCO design

to a digitally controlled oscillator (DCO), thus creating a gated-DCO. The 3-stage DCO converts the digital value to a frequency proportional to W[0:4]. Each stage of the DCO consists of a bank of parallel binary-sized inverters controlled by the digital value (W[0:4]) as shown in Fig. 6.5(b) and (c). The frequency of the DCO for the i^{th} word (W_i), ignoring second-order effects such as non-linearity, is given by:

$$F_{W_i} = W_i * F_0 \tag{6.5}$$

where F_0 is the unit frequency of the DCO corresponding to a $code1$ when $W = 00001$. The clock to the counter is driven by the DCO, and the enable signal is controlled by the pulse width (T_{p_i}). Hence, the counter output is given by:

$$DT_i = F_{W_i} * T_{p_i} = W_i * F_0 * T_{p_i} \tag{6.6}$$

From Eq. 6.6 we can observe that the counter output is proportional to the product of the weigth (W_i) and the pulse width (T_{p_i}) of the gating signal. This is a true mixed-signal multiplier, where one of the input is a pulse (analog), and the other input is a digital value stored in the local memory. The counter provides accumulation natively. In the current design, all the pre-synaptic neurons (fan-ins) send pulses (T_{p_i}) one after another.

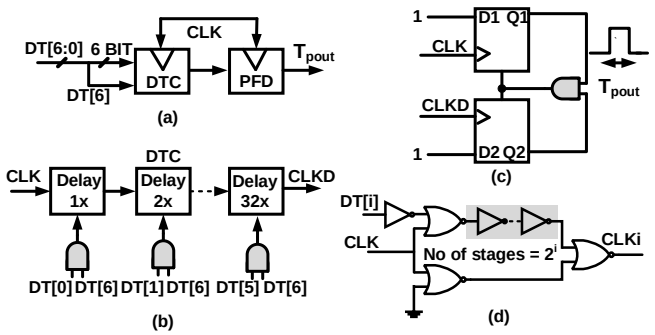


Figure 6.6: (a) Circuit implementation of the digital-to-pulse converter (DPC) illustrating the (b) cascade of delay elements, (c) phase-frequency Detector (PFD) and (d) the delay cell for DT[i] input.

For each of these pulses, the TD-MS MAC loads the corresponding synaptic weight (W_i) and computes $W_i * T_{p_i} * F_0$. For N such pre-synaptic pulses, at the end of an update, the counter output represents:

$$DT = \sum_{i=0}^{N} DT_i = \sum_{i=0}^{N} W_i * T_{p_i} * F_0 \tag{6.7}$$

Before the beginning of the update the counter is reset to all zeros expect for the MSB which is reset to 1. This allows the counter to count up (for $W_i > 0$) or count down (for $W_i < 0$). At the end of the update cycle, the sign of the MSB indicates whether $DT > 0$ or $DT < 0$, a key threshold that is subsequently used by the non-linear activation function, that is discussed in the next section.

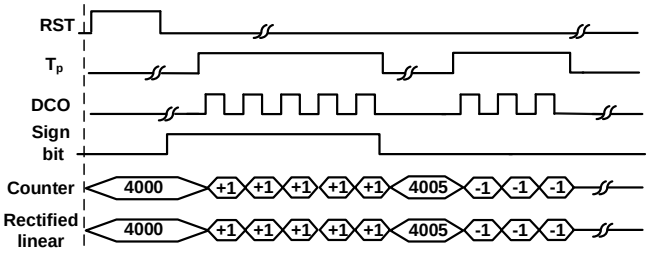


Figure 6.7: Timing diagram for the time-domain MAC logic illustrating accumulation over two cycles, one with positive synaptic weight and one with a negative synaptic weight.

6.3.2 Digital to Pulse Converter (DPC) Based Non-Linear Activation

After the linear MAC operation, a digital-to-pulse converter (DPC) is used to implement the non-linear activation function of the neuron. The schematic diagram of the DPC is shown in Fig. 6.6(a). It consists of a digital-to-time converter (DTC) followed by a phase-frequency-detector (PFD). It receives a 7bit input (DT) which is obtained from the output of the TD-MS MAC. The DTC consists of cascaded delay chains, as shown in Fig. 6.6(b). Fig. 6.6(c) shows the schematic of the PFD. The PFD receives clock signal (CLK) and a delayed version of CLK (CLKD) as inputs. Each delay chain consists of two paths: one path directly feeds the input to the output through NOR-based buffers, and another path transmits the input through a bank of binary-sized inverters, as shown in Fig. 6.6(d). The buffers are sized to produce binary-weighted delays that scale as 2^i for i=0 to 6. The control word for the DPC is DT (output of the MAC), whose MSB is 1 when the accumulated value of the MAC is non-negative or 0 otherwise. We use the MSB to gate all the lower bits before they are applied as a control word to the DTC, as shown in Fig. 6.6(b). As a result, the DPC produces an output given by:

$$T_{p_{out}} = \begin{cases} 0 & \text{if } DT \leq 0 \\ DT * T_0 & \text{if } DT > 0 \end{cases} \tag{6.8}$$

This shows that the final output ($T_{p_{out}}$) of the TD-MS neuron implements an ReLU activation function:

$$T_{p_{out}} = ReLU(DT) \tag{6.9}$$

To generalize, in a cascaded design where the output of one pre-synaptic neuron, i, drives the input of the post-synaptic neuron, j, the time-domain pulses, T_{pij} represent propagation of information through the network. In ML literature, this is typically denoted by X_{ij}. Here $T_{pij} = X_{ij} * T_0$. The synaptic weight between the two neurons is W_{ij}. By combining, Equations 6.7 and 6.8, we can write down the output of the j^{th} neuron ($T_{p_{outj}}$)

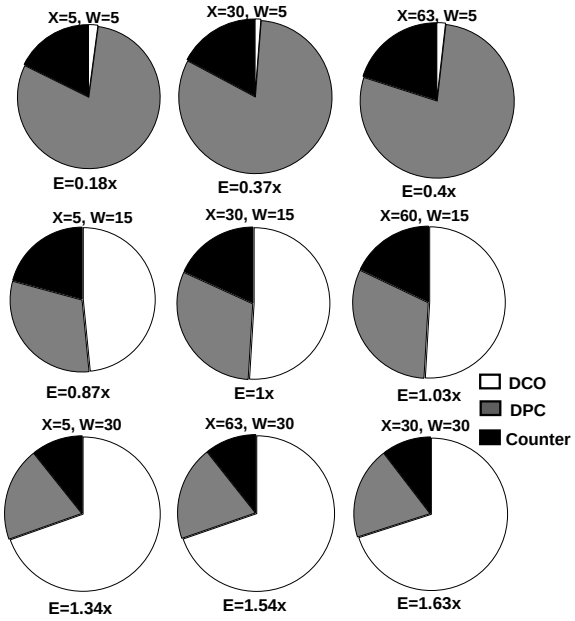


Figure 6.8: Break-down of the energy to compute for TD MAC across various input and weight codes. The energy is normalized with respect to a centrally placed code.

simply as:

$$T_{p_{outj}} = T_0 * ReLU\left(\sum_j W_{ij} * X_{ij} * F_0 * T_0\right) \tag{6.10}$$

Equation 6.10 illustrates that the TD-MS neuron can successfully implement an ReLU based neuron and encodes both the inputs from pre-synaptic neurons and output in time-domain. It should be noted that the portion of the equation within the parenthesis is dimensionless. An example of the timing diagram is shown in Fig. 6.7, where the cascaded operation of the DCO, the counter based accumulator and the output of the ReLU are shown with specific values. The treatment, shown above, captures first order effects only. Second order effects, such as non-linearity of DCOs lead to error in calculating the MAC, which can lead to loss of accuracy. However, RL algorithms like many other online ML techniques is forgiving to such non-idealities, albeit within limits.

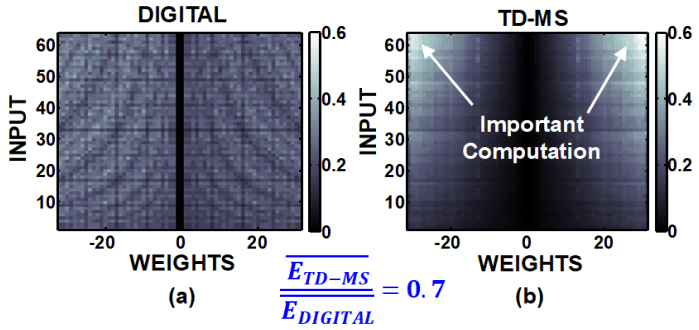


Figure 6.9: A 2D plot of the energy surface illustrating the energy/MAC for (a) digital implementation and (b) TD-MS design, with 6b inputs (X and W)at V_{CC} 0.6V. We note that more energy is consumed for important operations where $|X|$ and $|W|$ are large. The average energy/MAC for the TD-MS implementation is $0.7\times$ that of the digital implementation.

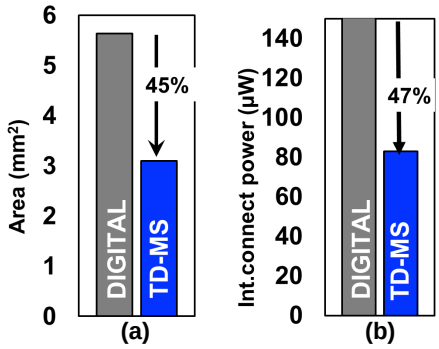


Figure 6.10: Results from synthesis and place-and-route of the complete design show that the TD-MS consumes: (a)45% lower area and (b 47% lower interconnect power, compared to a digital design.

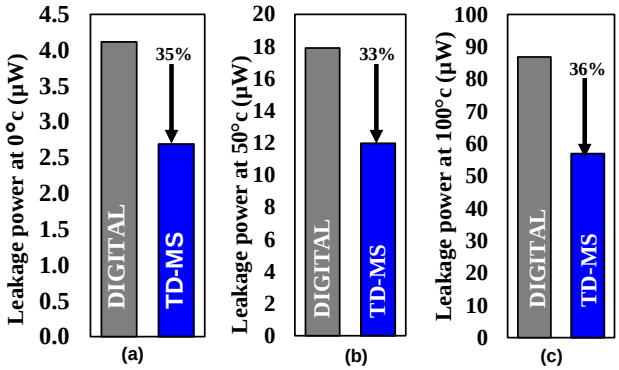


Figure 6.11: Simulated post-synthesis leakage power at three different temperatures (a) 0C (b) 50^oC (c) 100^oC for the digital and TD-MS design

6.3.3 Energy and Power Analysis of the Proposed TD-MS Design

For the application in hand, sensor data (state-estimates) is acquired at regular intervals. Hence, the amount of compute needed per unit time is approximately constant, depending on the speed of the mobile-robot and the complexity of the environment. Hence, we wish to compute at the lowest possible energy, while being cognizant of the range and resolution of the data. TD-MS designs provide some unique characteristics, as we describe here. From simulations on the 55nm technology node at $V_{CC} = 0.6V$, we explore the energy consumed by the various blocks (after synthesis with tight constraints on power and area) in the time-based neuron and plot them in Fig. 6.8. Since the number of switching events in the TD-MS design depend on the magnitude of the operands (X_{ij} and W_{ij}), the energy consumed also shows a strong dependence on X_{ij} and W_{ij}. The 9 pie charts (for W=5, X=5 to W=30, X=63) illustrate the normalized energy per MAC. For lower values of the DCO code ($W_{ij} < 15$) the energy consumption is dominated by the DPC. Subsequently, the DCO power becomes dominant when the DCO code increases beyond 15. Further with increasing magnitudes of the operands, the total energy to compute a MAC also increases. This is an interesting design choice, because of two observations specific to the algorithm. Firstly, larger operands correspond to more important parts of computation in the network and typically represent the flow of critical information. Secondly, in trained neural networks, the magnitudes of the weights tend to cluster around low values. We take advantage of both of these observations in TD-MS designs because of its energy-scalability with the importance of the computation, which is a bio-mimetic approach. We capture this in Fig. 6.9 by plotting the energy to compute as a function of the input (X_{ij}) and the weight (W_{ij}). In binary coded arithmetic, we do not see any such trend, since the number of switching events is not correlated to the magnitudes of X_{ij} or W_{ij}. It is worth noting that a digital design is expected to produce higher performance than a TD-MS design which inherently uses time for data-encoding. However, as long as the time required to complete the data-processing is acceptable, as is the case in this application, TD-MS provides a bio-mimetic

and energy-efficient approach. We note that on an average, where all the combinations of X_{ij} and W_{ij} are equally likely, TD-MS based MACs spend 30% lower energy at iso-V_{CC} compared to an array based digital MAC. We also compare the post-synthesis area of the TD-MS design vis-a-vis a digital design and observe (1) 45% lower system area and (2) 47% lower interconnect power. The lower interconnect power comes from the fact that the outputs of the neurons travel through single bit interconnects and undergo only one $1 \hookrightarrow 0$ and one $0 \hookrightarrow 1$ transition. This saves both leakage and switching power compared to a bit-parallel digital design, albeit at the cost of performance. This also consumes less dynamic energy compared to a bit-serial digital interconnect where a 6-bit word will typically undergo multiple transitions. The compact area of the TD-MS design is further reflected in Fig. 6.11 where the total simulated leakage power of TD-MS and digital designs across three temperatures 0^oC, 50^oC and 100^oC corners are shown. On an average the leakage power in TD MAC is 35% less compared to the digital implementation.

6.3.4 Time-Domain Stochastic Synapses

It is well studied in the ML community [71] that the introduction of stochasticity in synaptic connections avoid over-fitting of data. Along with stochasticity, drop connect is also used to eliminate random synaptic connections during training. Both of these regularization techniques assist in generalizing the learned neural network model to unknown environments. Here we introduce stochasticity in the synaptic connections in the time-domain, as shown in Fig. 6.12(a). High-speed local linear-feedback-shift-registers (LFSRs) drive multiplexers which tap various nodes of a synaptic buffer chain. A part of the buffer chain provides a fixed delay, and a part of the buffer chain has a stochastic delay. Stochasticity is introduced by changing the delay between the $1 \hookrightarrow 0$ and the $0 \hookrightarrow 1$ transitions. Fig. 6.12(b) depicts the LFSR circuit and how the multiple output bits enable stochasticity as signals propagate. Since these synaptic connections propagate X_{ij}, the variable delay introduces controlled randomness in the operand X_{ij}. Further for a fixed LFSR code, the multiplexer selects a 0,

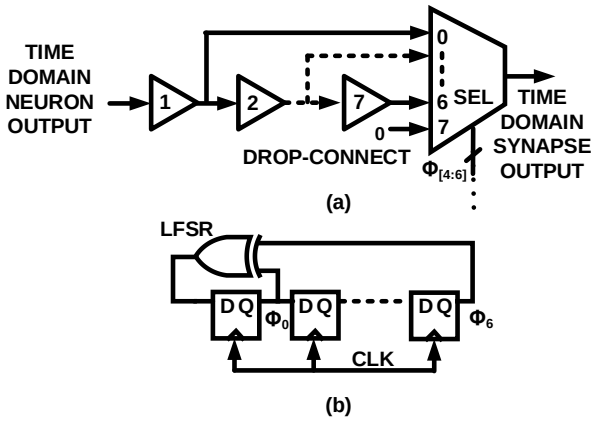


Figure 6.12: Stochasticity and drop-connect are implemented by intentionally introducing randomly varying delays between the $1 \hookrightarrow 0$ and the $0 \hookrightarrow 1$ transitions. The randomness is introduced by (a) an LFSR which drives the (b) MUX-select

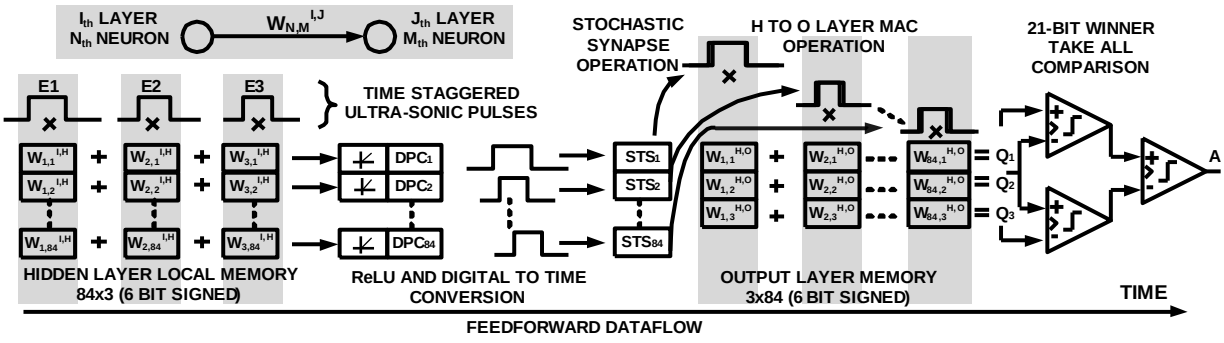


Figure 6.13: Timing diagram for the feed-forward neural network illustrating the parallel and sequential components of the data-flow

thereby preventing any signal propagation and enabling drop-connect.

6.3.5 Information Flow in the Feed-Forward Direction

The timing diagram for information flow in the feed-forward network is shown in Fig. 6.13. The input signals from the ultrasonic sensors (E_1, E_2 and E_3) are processed in a sequential manner. 84 TD-MS neurons perform MAC operations with weights ($W_{1,1}^{IH}$ to $W_{1,84}^{IH}$) and the E_1 pulse, all in parallel. Next for the E_2 pulse from the ultrasonic sensor, the next set of eighty-four weights ($W_{2,1}^{IH}$ to $W_{2,84}^{IH}$) are used. This is followed by the pulse from E_3. Next, the time-domain outputs of the 84 neurons in the hidden-layer are generated sequentially. The pulse generated by the first neuron of the hidden-layer is multiplied by $W_{1,1}^{HO}$, $W_{1,2}^{HO}$ and $W_{1,3}^{HO}$ in parallel at the three output neurons respectively. Next, the output

pulse of the second neuron of the hidden-layer is generated, multiplied by $W_{2,1}^{HO}$, $W_{2,2}^{HO}$ and $W_{2,3}^{HO}$ in parallel and accumulated. This continues till all the 84 neurons of the hidden-layer have fired. Finally, the outputs of the three output neurons are generated and they encode the Q-values corresponding to the three actions. A winner-take-all (WTA) decides on the most favourable action and sends a 2-bit command to the Respberry-PI, which in turn enables the corresponding motor controls. The test-chip runs on a system clock, which maintains synchronicity among sensor activation, data flow, memory read operations and output generation.

6.4 System Architecture for Learning via Back-propagation

In the previous section, we have described data flow in the forward direction. For online reinforcement-learning (RL) we perform back-propagation using gradient descent. Back-propagation involves the following discrete steps: (1) Evaluation of the output gradients (GD_O) (2) Evaluation of the gradients at the hidden layer (Δ_H) (3) Evaluation of the gradients for the weights from the input to the hidden layer (w^{IH}) and from the hidden to the output layer (w^{HO}) (4) Updating all the weights.

The output gradient is computed using a support vector machine (SVM) hinge-loss function. Fig. 6.14(a) illustrates the steps implemented in the current system. (GD_0) takes two cycles. Let us denote the Q-value corresponding to the label stored in the state-action table by Q_{Y_I}. Here Y_I is the action corresponding to maximum Q-value. Q_{Y_I} can be expressed as:

$$Q_{Y_I} = R_t + \gamma . \max(Q(S_{t+1}, A_t)) \tag{6.11}$$

Next, the SVM hinge-loss is evaluated as:

$$L_j = \sum_{j \neq Y_I} \max(0, Q_j - Q_{YI} + 1) \tag{6.12}$$

The gradient of loss for (Y_I) is calculated using [72]:

$$GD(Y_I) = \sum_{j \neq Y_I} ((Q_j - Q_{Y_I} + 1) > 0?1 : 0) \tag{6.13}$$

Finally, we calculate the gradient of loss for $(j \neq Y_I)$ as:

$$GD(j) = ((Q_j - Q_{Y_I} + 1) > 0?1 : 0) \tag{6.14}$$

It takes two cycles to calculate the gradient. Fig. 6.14(b) depicts the schematic of the corresponding circuit. Q_j is added with 1 and subtracted from Q_{YI}. The subtracted value is compared with 0. The comparator output is used as the select bit for selecting 0 or 1 in the accumulator. Hidden layer gradients are computed using the dot product between the output gradient (GD_0) and w^{HO}. This is illustrated in Fig. 6.14(c). It can be mathematically written as:

$$\Delta^{H,O} = GD_O * w_{I,J}^{H,O} \tag{6.15}$$

84 parallel units of DCOs and DPCs are used for calculating the gradients of the hidden layer. Fig. 6.15 shows how the gradient of the hidden to the output layer is calculated. The hidden layer output is generated from the feed-forward iteration and is multiplied with the output gradients to finally obtain the gradient of the hidden to the output layer weights, dW^{HO}.

$$dW^{H,O} = H * w_{I,J}^{H,O} \tag{6.16}$$

Next, the hidden layer gradients are passed through the gradient of the ReLU and multiplied with the input signals to compute dW^{IH}. The circuit which implements dW^{IH} is

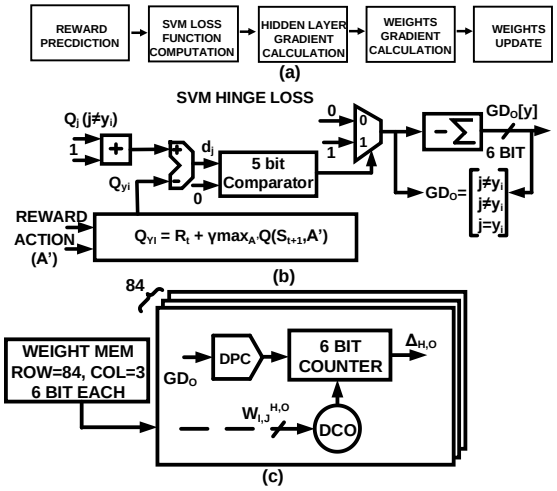


Figure 6.14: (a) The flow-chart showing the different components of back-propagation (b) Implementation of SVM hinge loss estimation during back-propagation (c) Schematic of the circuit that implements the update of the hidden layer gradients ($\Delta^{H,O}$)

schematically shown in Fig. 6.15. dW^{IH} is obtained as:

$$dW^{I,H} = X * ReLU_G(\Delta^{H,O}) \tag{6.17}$$

Finally, the update rules for w^{IH} and w^{HO} can be expressed as:

$$w^{H,O} = w^{H,O} - \frac{dW^{H,O}}{8} \tag{6.18}$$

$$w^{I,H} = w^{I,H} - \frac{dW^{I,H}}{8} \tag{6.19}$$

The design of hardware based back-propagation uses linear algebraic kernels which are similar to the feed-forward path. This allows us to share hardware resources. As a matter of fact, the 84 neurons at the hidden layer, consisting of DCOs and DPCs are fully programmable to compute both in the feed-forward (inference) as well as the feed-back (back-propagation and gradient-descent) directions.

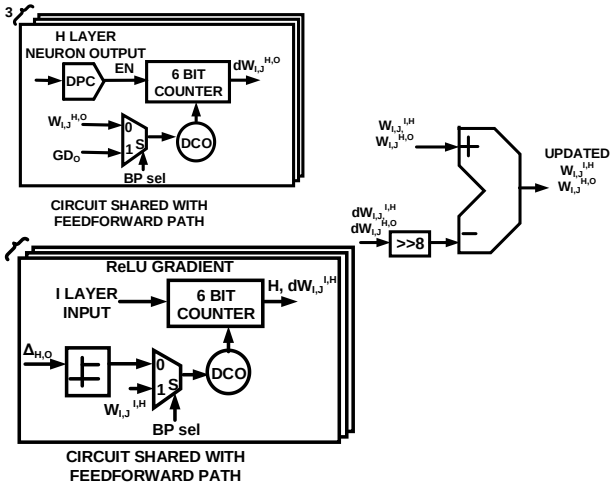


Figure 6.15: Circuit components showing the process of weight update across layers during back-propagation

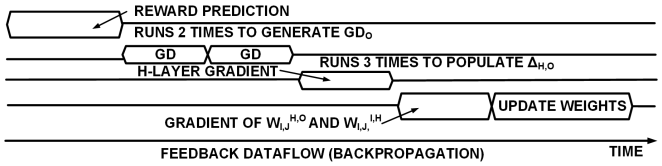


Figure 6.16: Timing diagram for back-propagation during reinforcement learning

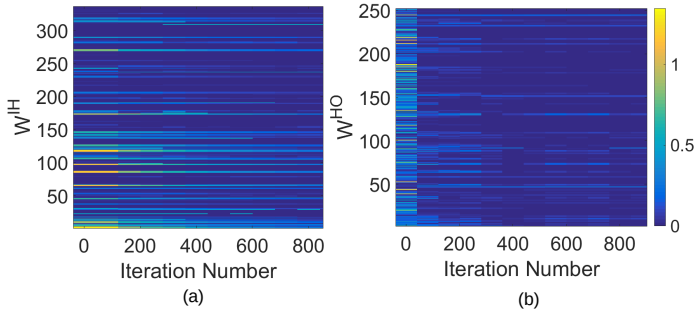


Figure 6.17: 2D surface plot illustrating the process of learning (weight update) for (a) the input to the hidden layer weights w^{IH} (b) the hidden to the output layer weights w^{HO}. The model weights are shown in column vectors and their evolution in time (number of iterations) is shown by color coding.

6.4.1 Data-Flow during Back-Propagation

Fig. 6.16 illustrates the timing diagram for performing back-propagation and gradient descent. The principal algorithmic steps have been shown here. After each update the corresponding Q-values and hidden layer weights are updated in each registers. Fig. 6.17 shows an example of how the weights are updated during reinforcement learning. Here the en-

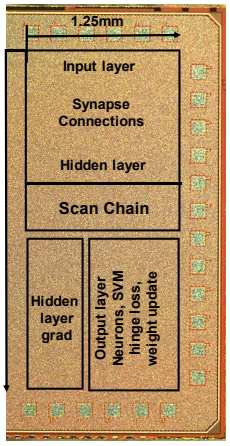
tire weight vector is shown as a vertical column and its evolution with time (or iteration number) is plotted. We note that as the system learns from its environment, the weights settle down to optimal value. It can also be noted that the magnitude of the majority of the weights is low, which has been discussed earlier in Section 6.3.3.

6.5 Design Flows for TD-MS Circuits

One of the main advantages of TD-MS circuits is our ability to map them to the traditional synthesis, and place-and-route (PnR) flows. As an example, we first design one stage of the DCO, characterize it and create the corresponding physical design. Next, we design the entire multi-stage DCO for target speed and dynamic range. One of the takeaways from the exercise was that the entire three stage DCO needed to be presented as a single block to the PnR flow, to avoid mismatches among the VCOs and timing-closure. Similarly, the DPCs are characterized as single macros and used in the back-end flows. By doing so, we minimize within-die variation, which results in better tracking of T_0 and F_0. Functional verification during synthesis is completed using behavioral RTL for the digital blocks, clock circuits and structurally modeling the DCO and DPCs. Once functional verification is complete, we combine the structural RTL with the rest of the digital blocks. This allows us to use back-end tools for the final design closure. TD-MS circuits use digital gates, which allow us to use both front-end and back-end tool-chains as long as the mixed-signal blocks synthesized and laid out as distinct macros.

6.6 Measurement Results

The test-chip is fabricated in 55nm CMOS process and occupies an active area of $1.25mm * 2.25mm$. The chip characteristics are shown in Fig. 6.18. Before going into the measurement results from the test-chip, let us define the application space. We use the test-chip in a mobile-robot that can perform obstacle avoidance in a limited space. The design is scalable and can be extended to more complex tasks and more complex obstacle maps. The reward



Chip Characteristics	
Technology	55nm 1P8M CMOS
Die area	1.25mm *2.5mm
Testing Interface	QFN package
Pin Count	24

Figure 6.18: Chip micro-graph and characteristics

function ($R(t)$) is defined as follows:

$$R_t = \begin{cases} -100 & \text{if the robot collides} \\ 30 - (d_{left} + d_{center} + d_{right}) & \text{otherwise} \end{cases} \tag{6.20}$$

Initially, the robot explores the environment by taking random steps and update the weights of its neural network (exploration phase). Progressively, the learning rate decreases and the robot harnesses its knowledge and experience to make optimal decisions on the

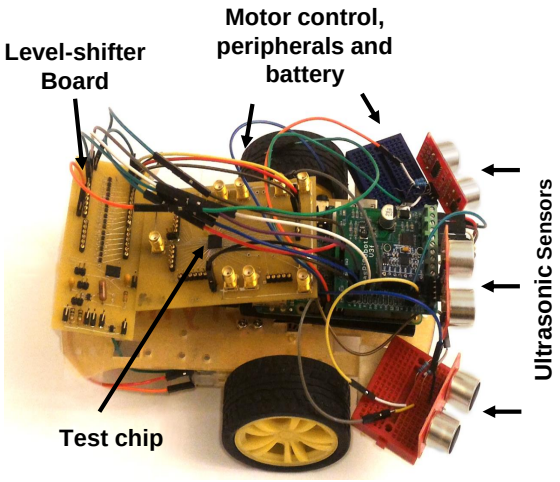


Figure 6.19: Overall system of the mobile-robot

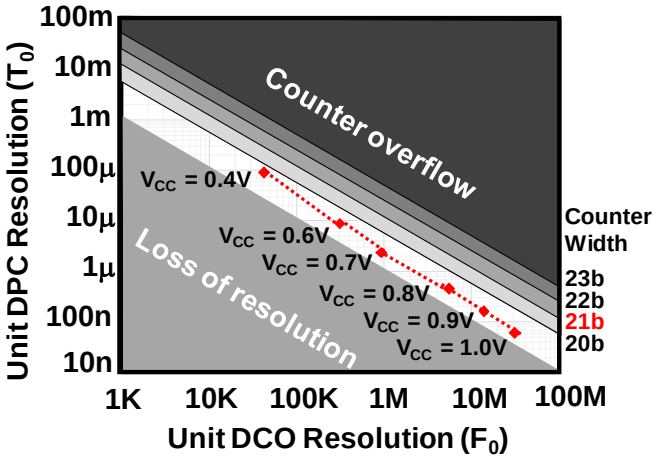


Figure 6.20: Design space of DCO and DPC showing scalability till 0.4V

fly (exploitation phase). Fig. 6.19 depicts the system setup with the test-chip along with the Raspberry PI and the motor drivers. We use an external MEMS oscillator for clock generation on the board. Level-shifters are used at the interface between the test-chip (1V supply) and the Raspberry PI (3.3V supply).

To functionally characterize the test-chip, we need to understand how the design space is constrained by the two fundamental design parameters, F_0 and T_0. This is shown in Fig. 6.20. If the DCO runs too fast (high F_0), the counters overflow. If the DCO runs too slowly (low F_0), then we lose resolution in the MAC units. For the current design, we choose a 21b counter, and the measured results show that the F_0 and T_0 track each other over a wide range of supply voltages. This provides correct operation as well as voltage scalability from 1.0V down to 0.4V. We suspect that the test-chip ceases to function at 0.4V, when the local register and memory circuits start to fail.

The measured frequency of the DCO, F_{DCO} (of the hidden layer) shows peak performance of 780MHz (at 1V). Fig. 6.21 shows the measured linearity of the DCO. We measure a peak INL of 1.2 LSB (1.6 LSB) at 1V (0.4V) and a peak DNL of 1.5 LSB (2 LSB) at 1.0V (0.4V). Fig. 6.22 shows the measured linearity of the DPC. The measured T_{DPC} exhibits acceptable linearity with a peak INL of 1.3 LSB (1.6 LSB) at 1V (0.4V) and DNL

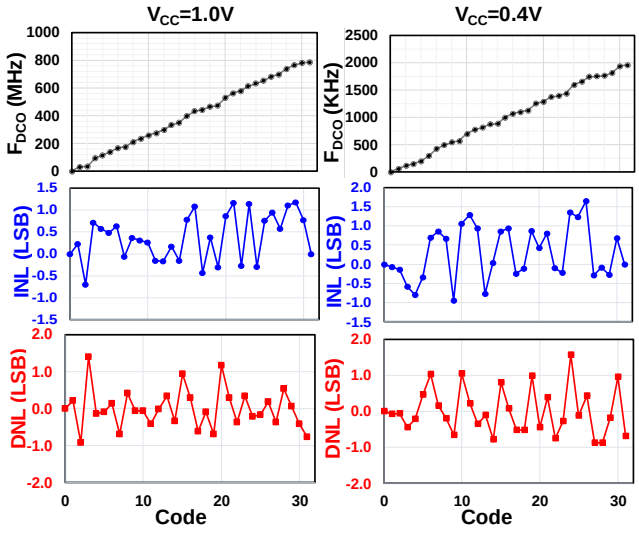


Figure 6.21: Measured linearity of the DCO

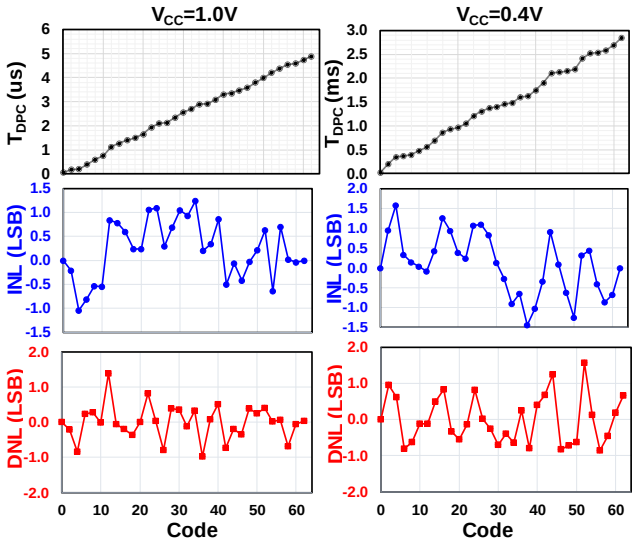


Figure 6.22: Measured linearity of the DPC

of 1.5 LSB (at 1V) and 1.8 LSB (at 0.4V). The dynamic range (DR) of the DPC is 4.9μsec and 2.8$msec$ for supply voltages of 1V and 0.4V respectively. We note that: (1) the DCO and the DPC are both composed of programmable buffer chains and their delays track each other across V_{CC} and (2) the programmability of the DPC and the DCO depend on the number of stages of buffer delay, which results in high linearity. Fig. 6.23(a) shows the intentionally introduced LFSR jitter normalized to a mean pulse width. We can observe that the jitter variance is within 40% and can be further tuned by programming the fixed and

stochastic parts of the synaptic delay chains. To evaluate the effectiveness of the stochasticity in the network, we train the robot for obstacle avoidance with and without stochasticity. Fig. 6.23(b) illustrates the loss function over time, and we observe that compared to a deterministic network, a stochastic network converges faster. From inference measurements in different environments, we have further validated that the system generalizes better with a stochastic network and prevents over-fitting on the training environment.

Fig. 6.24(a) shows the measured the clock frequency (F_{MAX}) as a function of V_{CC} from 0.4V to 1V. Fig. 6.24(b) illustrates the corresponding energy measured separately for inference and training. At a supply voltage of 0.8V the energy consumption for training and inferences are 1.5nJ and 670pJ respectively. This corresponds to a peak energy-efficiency

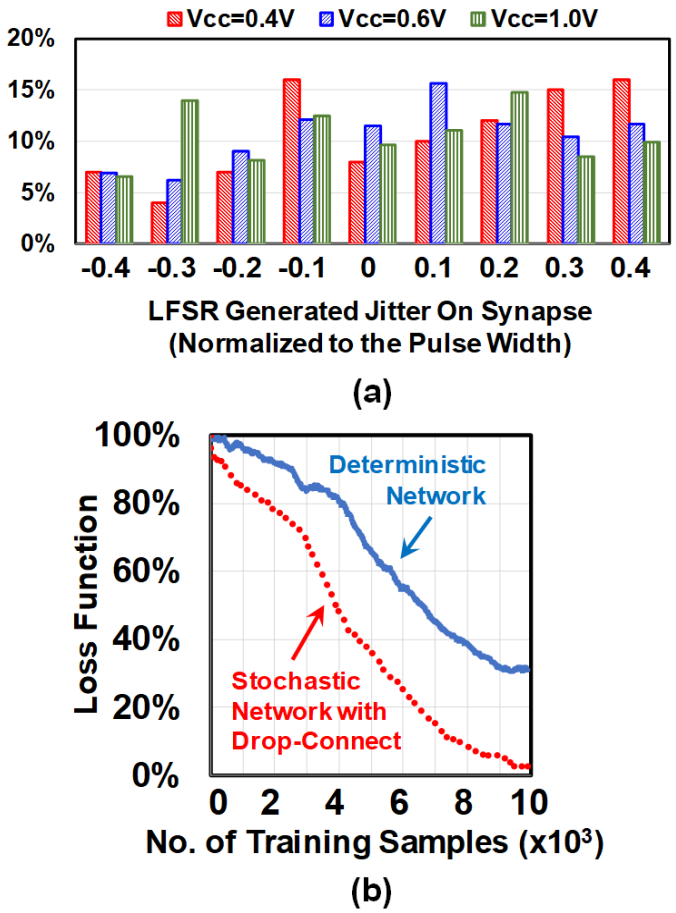


Figure 6.23: (a) The distribution of stochasticity on the synaptic connections as measured through the jitter on a mean synaptic pulse at V_{CC} of 0.4V, 0.6V and 1.0V. (b) Role of stochasticity on RL illustrating faster convergence compared to a deterministic network.

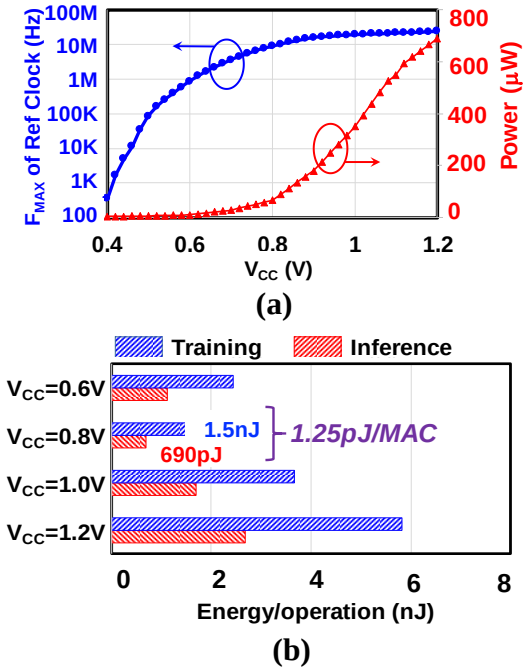


Figure 6.24: (a) Measured performance and power as a function of the supply voltage (b) Measured energy-efficiency as measured by the energy to perform a single inference and training on the network.

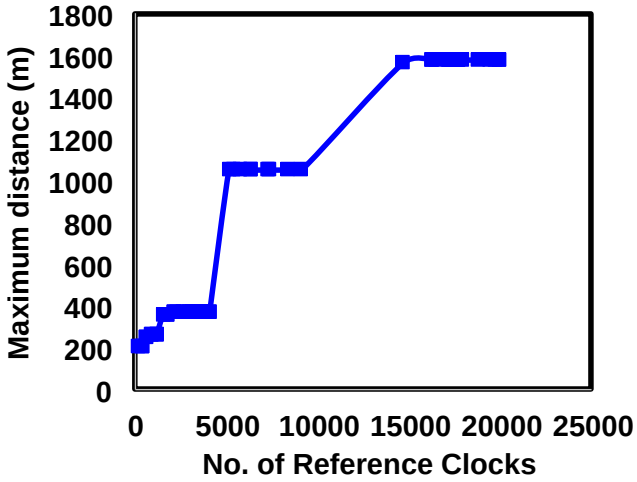


Figure 6.25: Distance covered by the robot via RL as a function of the number of clock cycles or iterations.

of 1.25pJ/MAC.

When the entire system is assembled, the supply voltages and power consumed by the various components are summarized in Table 6.1.

Table 6.1: Power consumed by different system components

Component	Current (A)	Voltage (V)	Power (W)
Ultrasonic sensor	0.015	5	0.075
Motor Drivers	1.2	6	7.2
Raspberry-PI	2.5	5	12.5
Test-chip	0.575m	1	0.69m

The Fig. 6.25 illustrates the distance moved by the robot in the presence of obstacles in a particular experimental setup.

In Table 6.2, we compare the current test-chip with state-of-the-art silicon implementations of neural networks. We believe that this is the first implementation of RL particularly for edge-robotics. Comparison with other reported results reveals that the TD-MS topology is energy-competitive and can work down to 0.4V at a peak power-efficiency of 3.12 TOPS/W.

Table 6.2: Comparison with the other hardware implementations of neural networks

	This work	[65]	[50]	[49]	[51]	[20]	[63]	[62]	[73]
ML System	Reinforcement Learning	Object Recognition	CNN-RNN	CNN	DNN	DNN	Spiking LCA	Spiking LCA	DNN
Technology	55nm	65nm	65nm	180nm	65nm	65nm	65nm	65nm	28nm
Circuit style	Time domain Mixed-signal	Digital	Digital	Digital	Digital	Digital	Analog	Digital	Digital
Area	$3.4mm^2$	$4mm^2$	$16mm^2$	$3.3mm^2$	$16mm^2$	$16mm^2$	$1.3mm^2$	$1.85mm^2$	$5.76mm^2$
Learning/ Training	Online in real time	Offline	Offline	Offline	Offline	Offline	Online	Online	Offline
Stochasticity	Present	Absent	Absent	Absent	Absent	Absent	Absent	Absent	Absent
Resolution	6b	16b	16b	4b-16b	16b	16b	NA	4b, 5b, and 16b	8b, 16b
Power	690μW at peak performance	121mW	63mW	7.5-300mW	45mW	278mW	87mW	268mW	63.5mW
Supply voltage	0.4-1V	1.2V	0.77-1.2V	Unavailable	1.2V	0.82-1.17V	0.9V	0.45-1V	0.6-1.1V
No. inferences/sec	254,000	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported	1.7M	9.9M	Not Reported
No. training/sec	118,000	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported
Performance (TOPS/W)	3.12	1.24	8.1	0.26-10	1.42	0.21	3.43	Not Reported	Not Reported
Application	Autonomous mobile-robotics	Object Recognition	General	Visual recognition	CNN processor	Vision	Vision	Vision	MNIST
Min. Energy/ inference	690pJ	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported	0.58μJ
Min. Energy/ training	1.5nJ	Not Reported	Not Reported	Not Reported	Not Reported	Not Reported	50.1nJ	Not Reported	Not Reported

CHAPTER 7
DEEP CONVOLUTIONAL NEURAL NETWORK TRAINING FOR EDGE-COMPUTING

Convolutional Neural Networks (CNNs) performs image classification. Generic CNN architecture is represented in Fig. 7.1. The input image is a 2D matrix format. The network has several sets of convolutional (Conv.) layers and fully connected (FC) layers. In FC layers every input neuron is connected to the output neuron through a parameter referred to as weight. FC layers are not feasible to be used for the large input image classifications. For images of size $224 * 224 * 3$, with the FC layer, the number of parameters for input layer will be 1.5M. If we choose 1000 hidden/output neurons, the total number of weight parameters will be 1.5G. If we use 2-byte for representation, the memory requirement blows up to 3GB. It will put huge area constraint on the hardware.

Therefore, convolutional layers are used to extract essential features. Convolution directly reduces the number of parameters required for computation. Convolution performs the local computation between input and filter. The filter slides across the input image/feature to produce output features. Generally, 10-100 filters are present in every layer. From Fig. 7.1 we can observe that primary convolutional layer output corresponds to feature like edge, change in color, etc. Primary set of convolutional layer represents Gabor filters. The intermediate and output convolutional layer output correspond to higher level features like glass, tire of the car. The output of the last convolutional layer is vectorized into a 1-dimensional vector. It is followed by 1 to 3 fully connected networks.

Table. 7.1 shows the details of different CNN architectures used. AlexNet, LeNet has few numbers of parameters required. Accuracy was tested for ImageNet database having 1.2M images. VGGNet and ResNet have comparable accuracy for 1.2M images. VGGNet has a more straightforward set of convolutional layers compared to ResNet. We choose

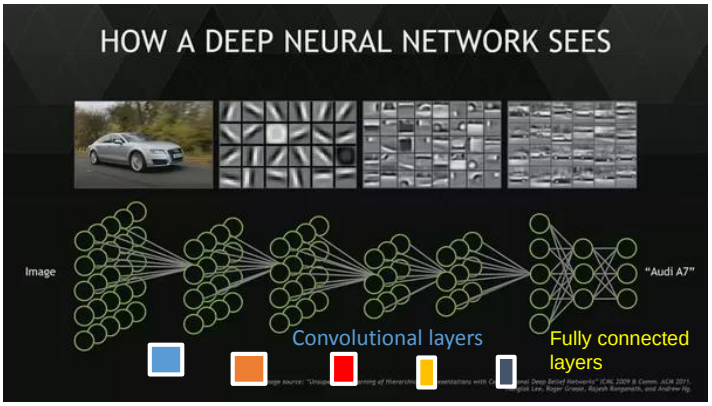


Figure 7.1: Deep Neural Network structure

Table 7.1: Summary of state of the art DNNs

Metrics	LeNet-5	AlexNet	VGG-16	ResNet-50
Top-5 error	N/A	16.4	7.4	5.3
Input Size	$28 * 28$	$227 * 227$	$224 * 224$	$224 * 224$
No. of conv. layers	2	5	16	49
No. of filters	6,16	96,384	64,512	64,2048
No. of weights	2.6k	2.3M	14.7M	23.5M
No. of FC layers	2	3	3	1
No. of weights	58k	58.6M	124M	2M
Total weights	60k	61M	138M	2M
Top-5 error(%)	NA	16.4	7.4	6.7

VGG-Net for transfer learning implementation for our work.

7.0.1 Need for transfer learning

VGG-Net has 13 convolutional and three fully connected layers. The total number of parameters for VGG-Net is 138M from Table. 7.1. If we use 2 bytes for storage, the total memory requirement will be 272MB. Training 138M in an area and power constrained edge-device will be not possible. Therefore, we choose transfer learning for implementing the on-learning for edge-devices. VGGNet [74] structure is shown in Fig. 7.2.

Transfer learning [75] is one of the promising solutions which helps to reduce the number of parameters required for training on edge computing systems. The TF learning struc-

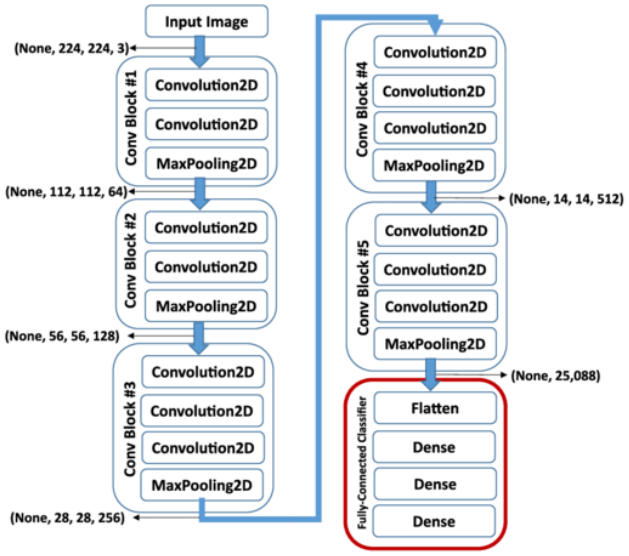


Figure 7.2: VGG-Net structure for image classification

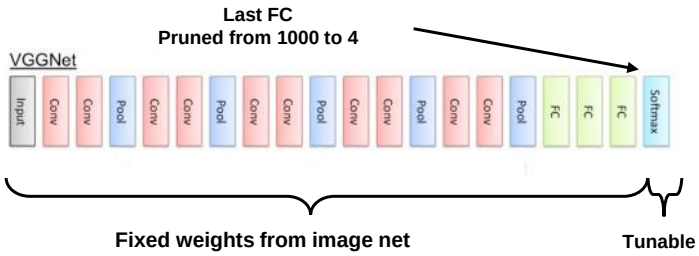


Figure 7.3: Transfer Learning for CNN

ture for VGG-Net is represented in Fig. 7.3. Pre-trained VGG-Net model is loaded for VGG-Net till the last FC layer. The last FC layer based on ImageNet has 1000 classes. We choose the Kaggle database having four classes to estimate the performance of VGG-Net loaded with ImageNet. Therefore, the number of classes is pruned from 1000 to 4 for Kaggle database. We train only the last FC layer for estimating the accuracy for Kaggle database. The classes present in the Kaggle database are dog, cat, horse, and humans.

Fig. 7.4 shows the methodology used for transfer learning. We use Keras library for deep neural network processing. The ImageNet trained model is loaded in Keras till the last fully connected layer. For the last FC layer training, we write a full custom code using python and Numpy. Numpy is a numerical algebra processing library in python. Through

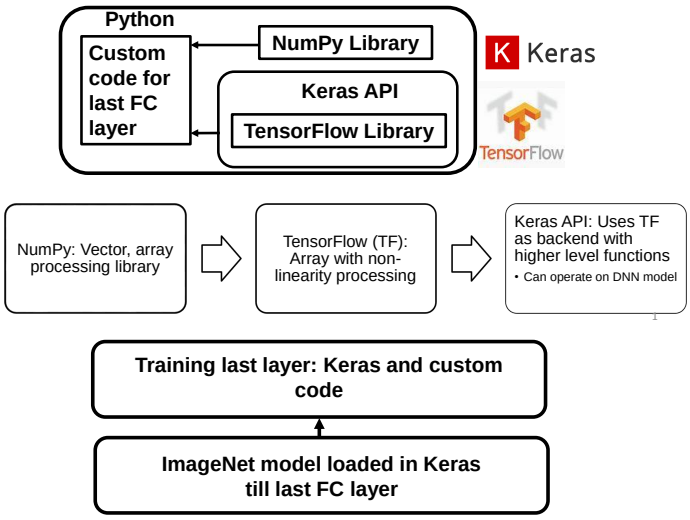


Figure 7.4: Methodology for Transfer Learning using CNN

this methodology, the performance of the transfer learning has been estimated.

Fig. 7.5 shows the last FC of VGG-Net corresponding to Kaggle database classes. The $(n-1)^{th}$ layer output has 4096 outputs (represented by matrix a). The 4096 outputs are multiplied with weights matrix of size $(4096 * 4)$ to get four output scores (z) which correspond to 4 classes namely cat, dog, horse, and humans. The output scores are converted probabilities, through softmax, and argmax functions.

The output scores are estimated by

$$z = a * w \tag{7.1}$$

The softmax classification estimates the probability of each score. The probability of each score is given by

$$p_i = \frac{e^{z_i}}{\sum_i e^{z_i}} \tag{7.2}$$

Computing the exponential function requires hardware to implement Taylor approximation for the exponential function. Softmax function is highly computationally expensive. Therefore, we choose the argmax function approximation for estimating the probabilities.

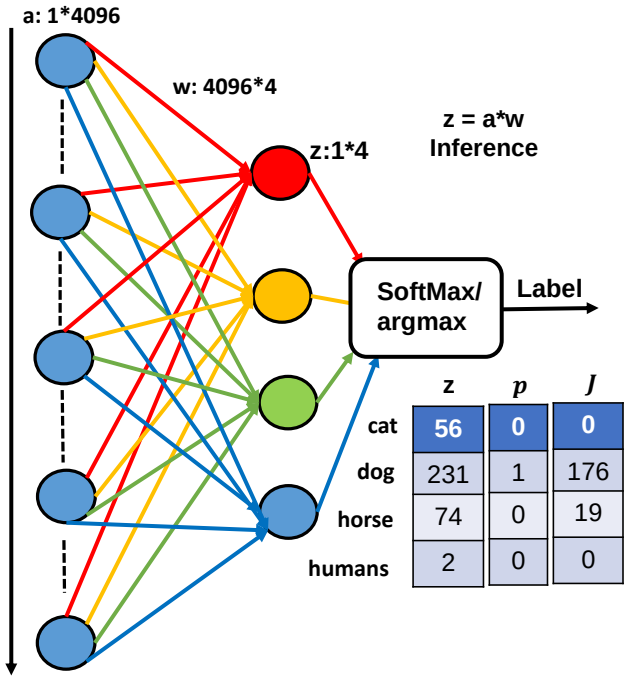


Figure 7.5: Inference and cost estimation for last FC layer

Argmax function squashes the output scores into 0's and 1's by comparison. The probability of the maximum score is 1. The probability of the corresponding to all other scores is 0. The argmax function is given by:

$$p_i = \max_i^N z_i \tag{7.3}$$

The cost function of the argmax function is defined by the hinge loss. The hinge loss is given by

$$J = \sum_{j \neq yi} \max(z_j - z_{yi} + 1) \tag{7.4}$$

From Fig. 7.5 we can observe that the training label is the cat. However, the score for the cat is less than the scores for horse and dog. Therefore, the loss concerning to dog label is the difference between dog and cat score plus 1. 1 is bias value added to make sure that correctly classified value should be at least one greater than the misclassified label.

Through, Eq. 7.4, the total loss corresponding to cat class is 198.

Training is an iterative process where we update the weight parameters to achieve correct classification. Transfer learning purpose we update the weights of the last FC network. Therefore, we need to find how the gradient of cost function concerning weights. The equations in the gradient descent are given by

$$\begin{aligned} \frac{\partial J}{\partial w_i} &= \frac{\partial z_i}{\partial w_i} \cdot \frac{\partial J}{\partial z_i} \\ \frac{\partial J}{\partial w_i} &= a^T \cdot \delta^L \end{aligned} \tag{7.5}$$

where $\frac{\partial J}{\partial w_i}$ is the gradient of the cost function with weights, $\frac{\partial z_i}{\partial w_i}$ is the gradient of the output scores with weights and δ^L corresponds to the gradient of the output layer.

δ^L for the argmax classification is given by

$$\begin{aligned} \delta_j^L &= ((z_j - z_{yi} + 1) > 0)?1 : 0 \\ \delta_{yi}^L &= -(\sum_{j \neq yi} \delta_j^L) \end{aligned} \tag{7.6}$$

where δ_{yi}^L is the gradient output corresponding to training label. δ_j^L is the gradient of output corresponds to all the non-training label scores. δ^L for the Softmax classification is given by

$$\delta_j^L = p_j - Yj \tag{7.7}$$

p_j is the probability of j^{th} class, and Yj is the one-hot encoded vector corresponding to the training label yi.

Table. 7.2 shows the output layer gradient (δ_j^L) for softmax and argmax classifiers. For misclassified label cat, softmax assigns -1 gradient whereas argmax assigns -2. Softmax assigns one gradient for the maximum score whereas argmax assigns score one each for labels whose score is greater than cat score. Negative gradient scales up the weight

Table 7.2: Gradient of output layer for softmax and argmax classifiers

Label	Score	δ^L (Softmax)	δ^L (Argmax)
cat	56	-1	-2
dog	231	1	1
horse	74	0	1
humans	2	0	0

corresponding to cat label. Positive gradient scales down the weight corresponding to the dog and horse labels (argmax). From simulation higher training accuracy for argmax is achieved due to this phenomenon.

The weight update is given by

$$w_i = w_i - \lambda \cdot \frac{\partial J}{\partial w_i} \tag{7.8}$$

Fig. 7.6 shows the iterative weight update procedure through gradient descent. The total number of training images is 320, the sample size of 32. $Samplesize = (Trainsize/batchsize) = 10$. Completing the weight update for 1 sample size is called one epoch.

7.0.2 Performance of ImageNet model for Kaggle database

ImageNet database has 1.2M images in its database. Images are organized in according with WordNet lexical database. Thousands of images depict each node of the hierarchy. Pre-trained model of the VGGNet on ImageNet model in the cloud is available for Keras. The last FC layer has 4Million parameters. The model provides probabilities for 1000 classes. Kaggle database has four classes. We feed in an image from the Kaggle database to ImageNet. If the top-5 probabilities of VGGNet output correspond to the input image label, the classification is considered to be correct. ImageNet doesn't explicitly have humans, cats, horse classes. If the output probabilities correspond to any of the cat-related classes, the classification is considered to be correct. Cat related classes are Egyptian, tabby cats, etc. The human-related classes are the suit, uniform, etc. We perform the logical or of the

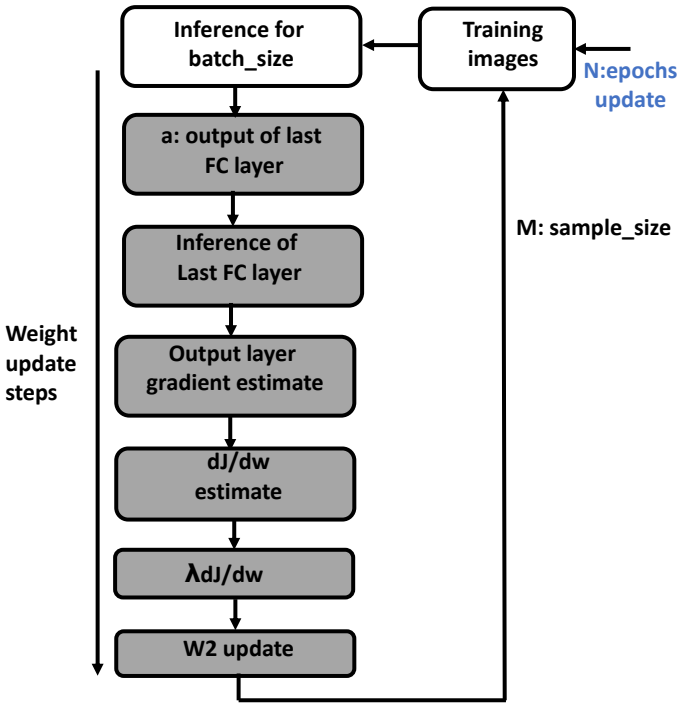


Figure 7.6: Iterative weight update procedure

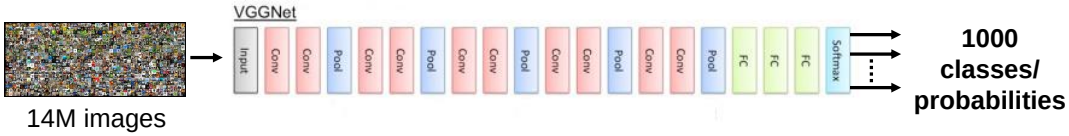


Figure 7.7: ImageNet model for VGGNet

Top-5 probabilities if the input label falls within any of the cat family the classification is considered to be correct.

Table. 7.3 shows the VGG-Net accuracy for Kaggle database. We can observe that it has less than 30% accuracy. It is 5% higher than the random guess. This experiment demonstrates the need for local training.

Fig. 7.8 shows the classification result of VGG-Net for horse image from Kaggle database. The Top-1% probability result corresponds to sorrel leaf, and Top-2% probability result corresponds to walker hound dog. The misclassified result by VGG-Net shows the need for local-training.

Table 7.3: VGG-Net accuracy for Kaggle database loaded with ImageNet database

	Accuracy
Top-1	29%
Top-2	37%
Top-5	44%

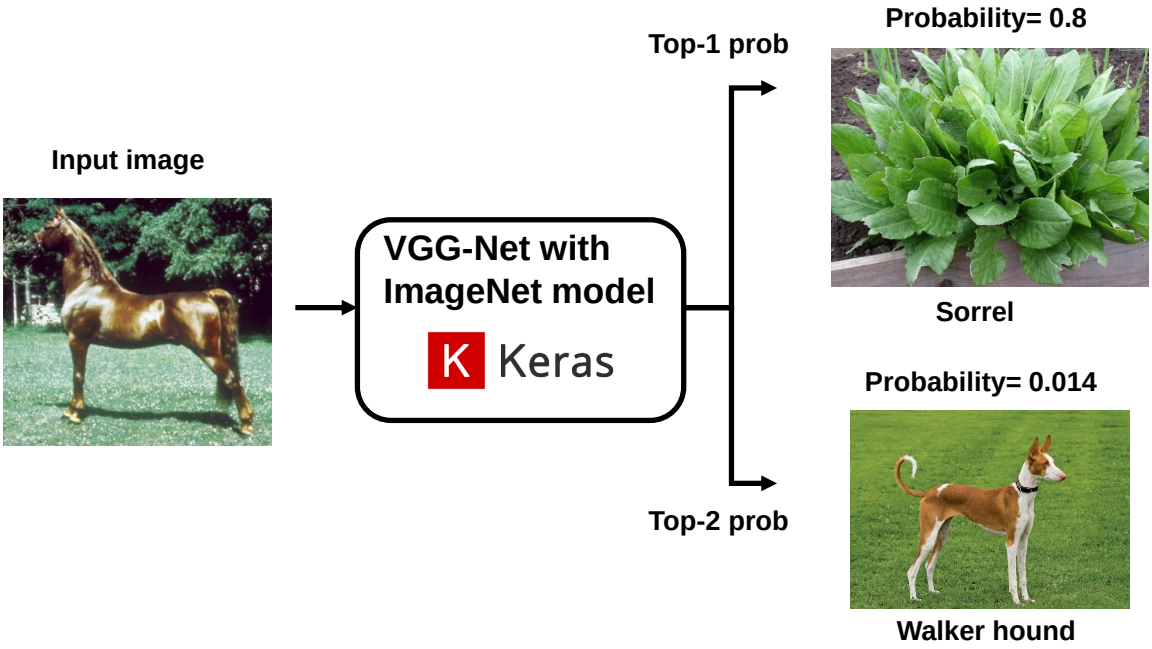


Figure 7.8: Classification for horse input from Kaggle database

Fig. 7.9 shows the loss vs. the number of iterations for Kaggle database. We can observe that the training error goes down within ten iterations itself.

Table. 7.4 shows the training and classification accuracy for softmax and argmax classifiers with VGGNet. The accuracy is more than 95%.

Fig. 7.10 shows the classification accuracy of the classifier with varying number of input samples. For larger input sample size the classification accuracy goes down. This is due to gradients of back-propagation becoming large. Small gradient steps are beneficial for weight update. However, the weight update drops down with more number of input samples. Reduced weight update reduces memory access and arithmetic operations.

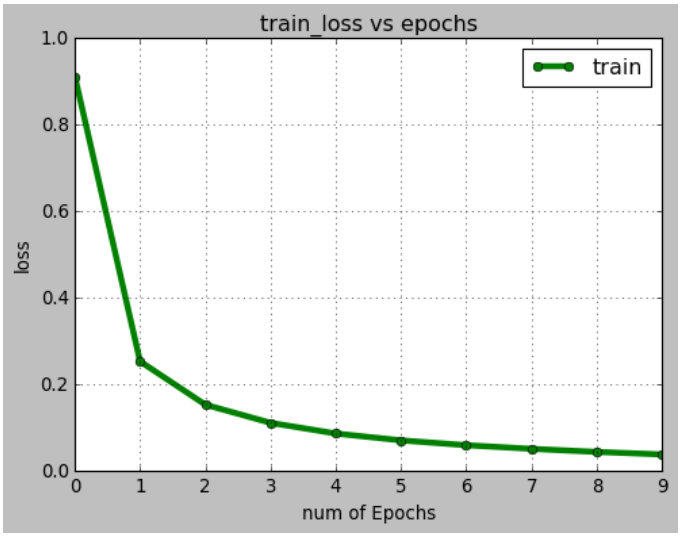


Figure 7.9: Training loss with number of iterations for VGG-Net trained with Kaggle database

Table 7.4: Accuracy for softmax and argmax classifiers

	Accuracy (Softmax)	Accuracy (Argmax)
Training	95.6%	100%
Top-2	95%	88.5%
Top-5	95.3%	94.25%

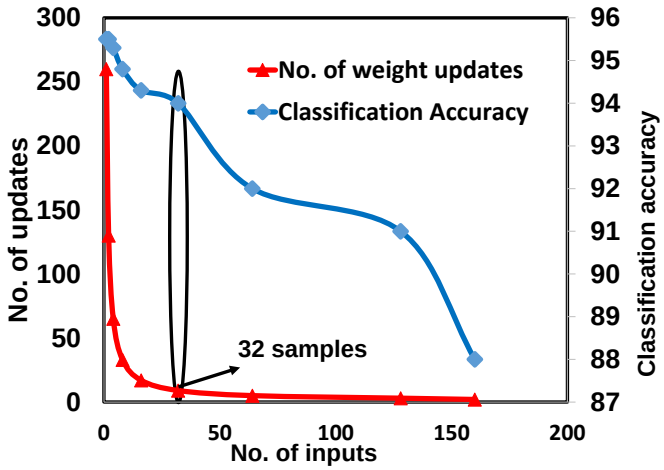


Figure 7.10: Classification accuracy vs No. of input samples

Therefore, we trade off 2% reduction of accuracy to get a 32X reduction in weight update.

Fig. 7.11 shows the proposed system for gradient descent update. a represents the fea-

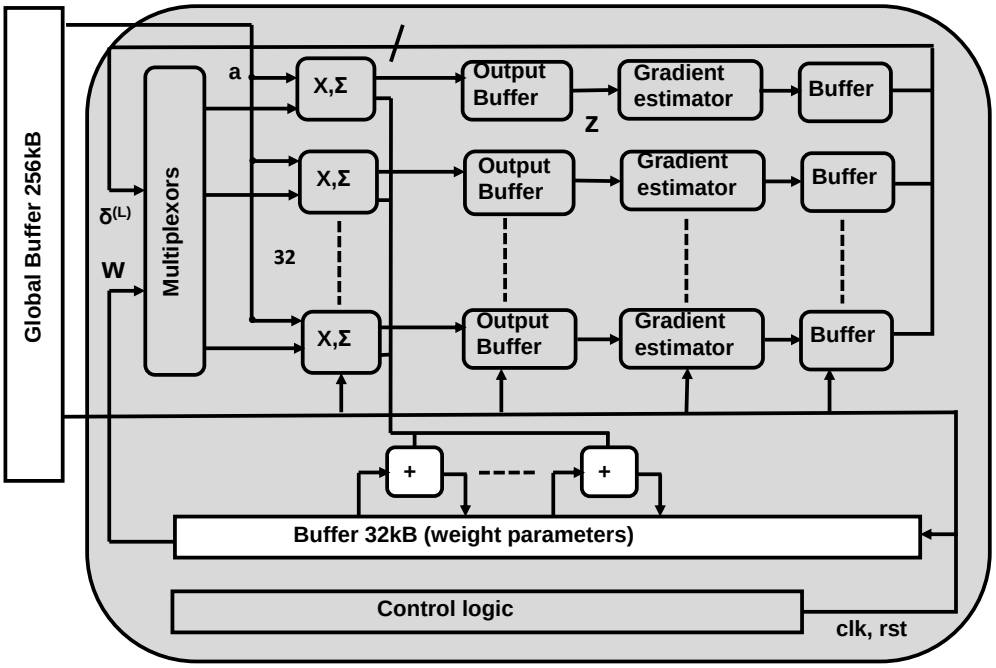


Figure 7.11: System Architecture for Proposed Gradient Descent Update

ture vector from $(n-1)^{th}$ hidden layer. a is multiplied by the weight vector to get the output scores. 32 ALUs process feature and the weight vector to produce output scores. The output scores are stored in buffer close to ALU. 32 gradient estimation units compute the gradient of the output layer through output scores and training labels. The estimated gradients are stored in gradient buffer. Gradient and weight are multiplexed used multiplexers which feeds the data to ALUs. Multiplexers are controlled by the state machine which generates control bit for inference and backpropagation. During back-propagation, the same feature matrix and gradient matrix are used to perform MAC operations. The output corresponds to $\frac{\partial J}{\partial w}$. $\frac{\partial J}{\partial w}$ is shifted by 10 bits to achieve scaling by 1024 which corresponds to the learning rate of 0.001. The shifted value is subtracted from weights to get a weight update. This corresponds to 1 epoch.

Fig. **??** shows the proposed 32 ALU processing feature matrix and weight matrices. It accesses 32 elements from 32 banks of feature (a). From the weight matrix every element is accessed sequentially $w_{1,1}$ to $w_{4096,1}$. After 4096 cycles the output scores ($z_{1,1}$ to $z_{4096,1}$) are

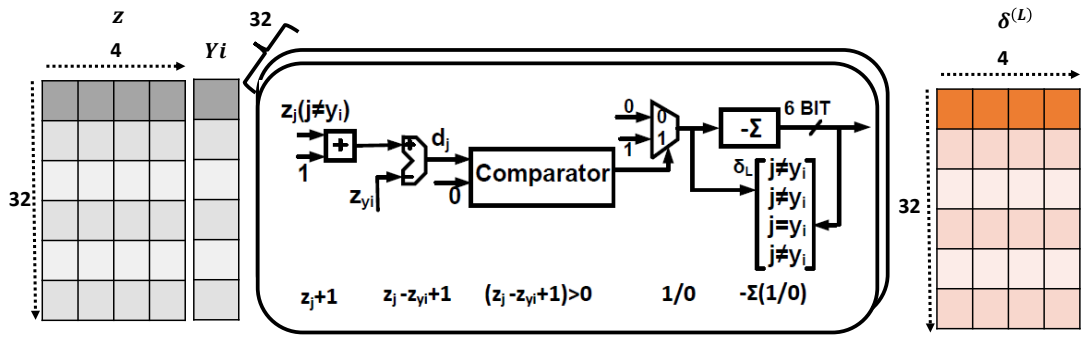


Figure 7.12: Gradient estimating circuit

written in output buffer. After writing the first column of the output, we write the second column scores after the next 4096 cycles. To evaluate all the four columns of output score $4096 * 4$ cycles are required.

Fig. 7.12 shows the circuit for output layer gradient estimation. Soon as the output scores are computed, they are passed through gradient estimating circuits. Gradient evaluating circuits needs the output scores (z) and training label (Yi). One is added to the score corresponding to non-training label and subtracted with the score corresponding the training label. If the result is greater than zero, then the gradient corresponding to the non-training label will be 1. Otherwise, it will be zero. It is implemented using mux and comparator circuits. The gradient corresponding to training label will be the negative sum of all gradients corresponding to non-training labels.

7.0.3 Control logic implementation

Control logic for inference is implemented using counters. There are two sets of counters implemented. The feature matrix from $(n-1)^{th}$ output layer is of size $32 * 4096$. Counter1 goes from 0 to 4095. Counter 1 is used access all the elements in 4096 rows sequentially. Second counter goes from 0 to 3. The second counter is used to access all the four columns of the weight matrix. For inference, we need $4096 * 4 = 16,384$ clock cycles. Gradient descent involves output layer gradient evaluation and weight update. The weight update

Table 7.5: Initial DC power simulations

Block	Power
Dynamic	49mW
Leakage	16mW
Total	65mW

Table 7.6: Comparison with traditional GPUs

	Proposed design	NVDIA Pascal	AMD Vega-10
Power	65mW	250W	225W
MAC/cycle	32	3584	4096
Type	14-b Fixed point	32b Floating point	32-Floating point
TOPS	0.16	9.3	13
$TOPS/W$	2.46	0.03	0.05

also requires 16,390 clock cycles.

7.0.4 Initial DC power simulations

Table. 7.5 shows the initial power from synthesized logic using DC compiler. The power is estimated at 1V supply, the temperature of (125^oC) and clock frequency of 2.5GHz. Dynamic power is high as expected. The total training time based on a 2.5GHz clock frequency is 13.1μsec. Power multiplied by time to perform one epoch gives 85μJ.

7.0.5 Comparison with GPU performance

Table. 7.6 shows the performance comparison of our proposed architecture with GPUs. Proposed design exhibits low $TOPS/W$ due to the limited number of MAC units. Higher $TOPS/W$ of the proposed design shows the advantage of edge computing.

CHAPTER 8

CONCLUSIONS

In this dissertation, techniques for performing low power inference and training for edge computing is presented. We present both system and circuit techniques for enabling intelligence for edge computing.

Chapter 3 describes the always-ON compressed system using compressive sensing. It is the one of the first ever reported gesture recognition system which was light powered. Smashed filter technique significantly reduces the data-movement to the processor and thereby reducing both computation requirement and energy spent during the memory read.

Chapter 4 describes the 64 multi-input time interleaved SAR ADC performing compressive sensing. It reduces the number of compressive measurements required by a factor of 2. Time-based compressive sensing is also described in Chapter 4.

Chapter 5 describes SAR and time-based vector multiplying ADCs. SAR-ADC uses low capacitor values of 50fF to achieve high energy efficiency. Even though time-based vector multiplying ADC has high INL, the data-driven technique helps to retain the same accuracy as compared to zero-INL ADC.

Chapter 6 describes time-based computing for reinforcement learning. We use a unique combination of sensor, MCU, accelerometer, and actuation are used to implement reinforcement learning. The time-based technique reduces the time required for explicit time to digital conversion. DPC based synapse significantly reduces the interconnect energy.

Chapter 7 describes the hardware micro-architecture, data movement and power estimation for batch gradient descent.

REFERENCES

[1] A. Amaravati, M. Chugh, and A. Raychowdhury, "A sar pipeline adc embedding time interleaved dac sharing for ultra-low power camera front ends," in *IFIP/IEEE International Conference on Very Large Scale Integration-System on a Chip*, Springer, 2015, pp. 131–149.

[2] W. Ran, "Real-time visual static hand gesture recognition system and its fpgabased hardware implementation," *ICSP*, no. 1, pp. 434–439, 2014.

[3] C.-T. Li, "A novel fpga-based hand gesture recognition system," *JCIT*, no. 1, pp. 221–229, 2012.

[4] D. Lee, "Vision-based remote control system by motion detection and open finger counting," *IEEE trans. on Consumer Electronics*, no. 1, pp. 2308–2313, 2009.

[5] B. Kellogg, V. Talla, and S. Gollakota, "Bringing gesture recognition to all devices," *Symposium on Networked System Design and Implementation*, no. 9, 2014.

[6] S Nayar, D Sims, and M Fridberg, "Towards self-powered cameras," *ICCP*, no. 9, 2015.

[7] J. Choi, S. Jungsoon, and D. Kand, "Always-on cmos image sensor for mobile and wearable devices," *JSSCC*, no. 9, 2016.

[8] B. Hu, F. Ren, Z.-Z. Chen, X. Jiang, and M. F. Chang, "An 8-bit compressive sensing ADC with $4GS/s$ equivalent speed utilizing self-timed pipeline sar-binary-search," *IEEE TCAS-II*, vol. 63, no. 10, pp. 934–938, 2012.

[9] M. Trakimas, R. D. Angelo, S. Aeron, T. Hancock, and S. Sonkusale, "A compressed sensing analog-to-information converter with edge-triggered SAR ADC core," *IEEE TCAS-I*, vol. 60, no. 5, pp. 1135–1148, 2013.

[10] W. Guo, Y. Kim, A. Sanyal, A. Tewfik, and N. Sun, "A single SAR ADC converting multi-channel sparse signals," *IEEE ISCAS*, vol. 60, no. 5, pp. 2235–2238, 2013.

[11] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: A survey," *Artificial Intelligence Review*, vol. 43, no. 1, pp. 1–54, 2015.

[12] M. Shoaib, K. H. Lee, and N. K. Jha, "A 0.16-107μW energy-scalable processor for directly analyzing compressively-sensed eeg," *IEEE TCAS-I*, no. 1, pp. 1105–1118, 2014.

[13] J. Zhang, Z. Wang, and N. Verma, "A matrix-multiplying ADC implementing a machine-learning classifier directly with data conversion," *IEEE ISSCC*, no. 1, pp. 332–334, 2015.

[14] E. Lee and S. Wong, "A 2.5GHz 7.7TOPS/W switched-capacitor matrix multiplier with co-designed local memory in 40 nm," *IEEE ISSCC*, no. 1, pp. 418–420, 2016.

[15] A. Anvesha, S. Xu, N. Cao, J. Romberg, and A. Raychowdhury, "A light-powered, always-on, smart camera with compressed domain gesture detection," in *Proceedings of the 2016 International Symposium on Low Power Electronics and Design*, ACM, 2016, pp. 118–123.

[16] S. Xu, A. Amaravati, J. Romberg, and A. Raychowdhury, "Appearance-based gesture recognition in the compressed domain," *ICASSP 2017*,
[17] V. Gruev and R. E. Cummings, "Implementation of steerable spatiotemporal image filters on the focal plane," *IEEE TCAS-II*, no. 4, pp. 332–334, 2002.
[18] R. Robucci, "Compressive sensing on a cmos separable-transform image sensor," *Proc. IEEE*, no. 1, 10891101, 2010.
[19] J. P. Slavinsky, "The compressive multiplexer for multi-channel compressive sensing," *Proc. IEEE ICASSP*, no. 1, pp. 3980 –3983, 2011.
[20] F. Chen, "Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors," *IEEE JSSC*, no. 1, pp. 3980 –3983, 2011.
[21] Y. Oike and A. E. Gamal, "Cmos image sensor with per-column adc and programmable compressed sensing," *IEEE JSSC*, no. 1, pp. 3980 –3983, 2013.
[22] M. A. Davenport, M. F. Duarte, M. B. Wakin, J. N. Laska, D. Takhar, K. F. Kelly, and R. Baraniuk, "The smashed filter for compressive classification and target recognition," in *Electronic Imaging 2007*, International Society for Optics and Photonics, 2007, 64980H–64980H.
[23] J. S. J. Choi and D.-S. Park, "A 45.5μw 15fps always-on cmos image sensor for mobile and wearable devices," *IEEE ISSCC*, no. 4, 114117, 2015.
[24] S. Wong, "Factorization for analog-to-digital matrix multiplication," *Report Standford*, no. 4, 114117, 2013.
[25] M. Gustavsson, "Cmos data converters for communication," *Kluwer Academic Publishers*, no. 4, 114117, 2000.
[26] ——, "Adc performance survey 1997-2015," *Kluwer Academic Publishers*, no. 4, 114117, 2000.
[27] D. Bankman, "An 8-bit, 16 input, 3.2pJ switched-capacitor dot product circuit in 28-nm cmos," *IEEE-Asian Solid State Circuits Conference*, no. 5, pp. 21–24, 2016.
[28] A. Anvesha, "A 130nm 165nJ/frame CD smashed-filter based mixed-signal class. for smart cameras," *IEEE Transactions on Circuits and Systems-II*, April 2017.
[29] T. Gerry, "A mostly-digital variable-rate continuous-time delta-sigma modulator adc," *JSSC*, 2010.
[30] K. Reddy, "A 16-mw 78-db sndr 10-mhz bw ct adc using residue-cancelling vco-based quantizer," *JSSC*, December 2012.
[31] R. G. Baraniuk and M. B. Wakin, "Random projections of smooth manifolds," *Foundations of computational mathematics*, vol. 9, no. 1, pp. 51–77, 2009.
[32] J. M. Carmona and J. Climent, "A performance evaluation of hmm and dtw for gesture recognition," in *Iberoamerican Congress on Pattern Recognition*, Springer, 2012, pp. 236–243.
[33] M. Müller, "Dynamic time warping," *Information retrieval for music and motion*, pp. 69–84, 2007.
[34] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern Recognition*, vol. 44, no. 3, pp. 678–693, 2011.

[35] Y. Shi, "An fpga-based smart camera for gesture recognition in hci applications," *Proceedings of the 8th Asian conference on Computer vision*, no. 1, pp. 718–727, 2007.

[36] Y Chen, B Luo, G Chen Y. Land Liang, and X. Wu, "A real-time dynamic hand gesture recognition system using kinect sensor," *IEEE Conference on Robotics and Biomimetics*, vol. 44, no. 3, pp. 2026–2030, 2015.

[37] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, 2006.

[38] C. C. Lee and M. Flynn, "A sar-assisted two-stage pipeline adc," *JSSC*,

[39] Y. Zhu, "A 50-fj 10-b 160-ms/s pipelined-sar adc decoupled flip-around mdac and self-embedded off. cancellation," *IEEE JSSC*, vol. 47, no. 11, 2012.

[40] Y. K. W. Guo and N. Sun, "A single sar adc converting multi-channel sparse signals," *ISCAS*,

[41] A. Bermak, "A 64 fj/step 9-bit sar adc array with forward error correction and mixed-signal cds for cmos image sensors," *TCAS-I*,

[42] C. Diaz, "Cmos tech. for ms/rf soc," *IEEE Trans. Electron Dev.*, vol. 50, no. 3, 2003.

[43] Z. Lin, "Modeling of capacitor array mismatch effect in embedded cmos cr sar adc," *Int. Conf. ASICs*, 2005.

[44] D. Bankman and B. Murmann, "Passive charge redistribution digital-to analog multiplier," *IEEE EL*, no. 1, pp. 386–388, 2015.

[45] D. Gangopadhyay and D. J. Allstot, "Compressed sensing analog front-end for bio-sensor applications," *IEEE JSSC*, vol. 49, no. 2, pp. 426–438, 2014.

[46] C Cortes, ""support-vector networks," *Machine Learning*, vol. 43, no. 1, 273–279, 1995.

[47] S Joshi, "$2pj$/mac 14b 88 linear transform mixed-signal spatial filter with 84db interference suppression," *IEEE ISSCC*, 2017.

[48] Y.-H. Chen, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE International Solid State Circuits Conference*, 2016.

[49] B Moons, "Envision: A 0.26-to-10TOPS/W sub word-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI," *IEEE International Solid State Circuits Conference*, 2017.

[50] D. Shin, "DNPU: An 8.1TOPS/W Re-configurable CNN-RNN processor for general-purpose deep neural networks," *IEEE International Solid State Circuits Conference*, 2017.

[51] J Sim, "A 1.42TOPS/W deep convolutional neural network recognition processor for intelligent IoT systems," *ISSCC*, 2016.

[52] P. N. Whatmough, "Dnn engine: A 28-nm timing-error tolerant sparse deep neural network processor for iot applications," *IEEE Journal of Solid State Circuits*, vol. 53, no. 9, September 2018.

[53] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[54] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[55] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York, NY: Springer New York, 2009, pp. 485–585, ISBN: 978-0-387-84858-7.

[56] R. Sutton, "Temporal credit assignment in reinforcement learning," *PhD thesis*, 1984.

[57] V. Mnih, "Playing atari with deep reinforcement learning," *NIPS Deep Learning Workshop*, 2013.
[58] D. Bankman, "An always-on 3.8μJ/86% CIFAR-10 mixed-signal binary cnn processor with all memory on chip in 28nm CMOS," *ISSCC*, 2018.
[59] J. Lee, "UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," *ISSCC*, 2018.
[60] C. Z. Xiaofan Lin and W. Pan, "Towards accurate binary convolutional neural network," *Computer Science, arXiv*, vol. 2, no. 5, pp. 1–14, 2018.
[61] J. Zhang, "A machine-learning classifier implemented in a standard 6t sram array," *Symposium on VLSI Circuits*, 2016.
[62] J. K. Kim, "A 640M pixel/s 3.65mW sparse event-driven neuromorphic object recognition processor with on-chip learning," *Symposium of VLSI Circuits*, 2015.
[63] F. N. Buhler, "A 3.43TOPS/W 48.9pJ/Pixel 50.1nJ/Classification 512 analog neuron sparse coding neural network with on-chip learning and classification in 40nm CMOS," *Symposium of VLSI Circuits*, 2017.
[64] A. Anvesha, S. Xu, J. Romberg, and A. Raychowdhury, "A 65nm compressive-sensing time-based adc with embedded classification and inl-aware training for arrhythmia detection," *IEEE-Bio-medical Circuits and Systems*, pp. 1–4, 2017.
[65] J Park, "Online RL NoC for portable hd object recognition processor," *CICC*, 2012.
[66] G. Cauwenberghs, "A nonlinear noise-shaping delta-sigma modulator with on-chip reinforcement learning*," *Analog Integrated Circuits and Signal Processing*, vol. 18, no. 2, pp. 289–299, 1999.
[67] A. Amravati, S. B. Nasir, S. Thangadurai, I. Yoon, and A. Raychowdhury, "A 55nm, 1.0V-0.4V, 1.25pJ/MAC time-domain mixed-signal neuromorphic accelerator with stochastic synapses for reinforcement learning in autonomous nano-robots," *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 124–126, 2018.
[68] K. Doya, "Complementary roles of basal ganglia and cerebellum in learning and motor control," *Current Opinion in Neurobiology*, March 2001.
[69] D. Poole, "Artificial intelligence: Foundations of computational agents," *Statistical Processing*, 2017.
[70] R. Bellman, "Dynamic programming," *Princeton University Press*, 2003.
[71] E. O. Neftci, "Stochastic synapses enable efficient brain-inspired learning machines," *Frontiers in Neuroscience*, 2016.
[72] Y. Tang, "Deep learning using linear support vector machines," *ICML*, 2013.
[73] P. N. Whatmough, "A 28nm SoC with a 1.2GHz 568nJ/Prediction sparse deep-neural-network engine with > 0.1 timing error rate tolerance for IoT applications," *ISSCC*, 2017.
[74] K. Simonyan, "Very deep convolutional networks for large-scale image recognition," *ICLR*, 2014.
[75] P. L, "Machine learning - special issue on inductive transfer," *Springer*, 2017.